VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Mobilní aplikace pro diagnostiku provozu IQMESH sítě

Mobile Application for Diagnosing of IQMESH Network Traffic

2018 Bc. Jan Kubala

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání diplomové práce

Student: **Bc. Jan Kubala**

Studijní program: N2649 Elektrotechnika

Studijní obor: 2612T041 Řídicí a informační systémy

Téma: Mobilní aplikace pro diagnostiku provozu IQMESH sítě
Mobile Application for Diagnosing of IQMESH Network Traffic

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Popis technologie IQRF a platformy Raspberry Pi.
2. Popis technologií pro zpracování a vizualizaci měřených dat.
3. Návrh a realizace IQRF/Wi-Fi brány na platformě Raspberry Pi.
4. Návrh a realizace IQRF/Bluetooth brány na platformě Raspberry Pi.
5. Sestavení prototypové aplikace pro sběr dat a ovládání komponentů využívajících technologii IQRF.
6. Zpracování a vizualizace měřených dat prostřednictvím vybrané technologie.
7. Zhodnocení výsledků a závěr.

Seznam doporučené odborné literatury:

[1] ALLEN, Grant. *Android 4: průvodce programováním mobilních aplikací.* 1. vyd. Brno: Computer Press, 2013, 656 s. ISBN 978-80-251-3782-6.
[2] VÁVRŮ, Jiří a Miroslav UJBÁNYAI. *Programujeme pro Android. 2., rozš. vyd.* Praha: Grada, 2013, 250 s. Průvodce (Grada). ISBN 978-80-247-4863-4.
[3] JIRKA, Jakub. *Moderní informační technologie pro řízení. Učební text. 1.vyd.* Ostrava: VŠB-TU Ostrava, 2012.
[4] MICRORISC s.r.o. *IQRF Quick Start Guide For IQRF OS v3.07D and higher.* 2017. [online] Dostupné z: http://www.iqrf.org.
[5] MICRORISC s.r.o. *IQRF OS Operating System Version 4.00D for (DC)TR-7xD - User's Guide.* 2017. [online] Dostupné z: http://www.iqrf.org.
[6] MICRORISC s.r.o. *IQRF DPA Framework Technical Guide v3.00.* 2017. [online] Dostupné z: http://www.iqrf.org.
[7] MICRORISC s.r.o. *IQRF Cloud Technical guide.* 2015. [online] Dostupné z: http://www.iqrf.org.
[8] MICRORISC s.r.o. *SPI Implementation in IQRF For (DC)TR-7xD.* 2016. [online] Dostupné z: http://www.iqrf.org.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Martin Pieš, Ph.D.**

Datum zadání: 01.09.2017

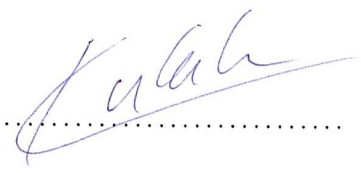
Datum odevzdání: 30.04.2018

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 27. dubna 2018



.....

Rád bych na tomto místě poděkoval vedoucímu práce, panu Ing. Martinovi Piešovi, Ph.D., za cenné rady a připomínky, které mi poskytl při řešení této práce. Dále bych chtěl poděkovat své přítelkyni a mé rodině za podporu a trpělivost.

Abstrakt

Tato práce se nejprve zabývá technologií IQRF. Poté popisuje počítač Raspberry Pi 3 a jeho využití v rámci IoT. V dalších kapitolách popisuje princip komunikačních technologií Bluetooth a Wi-Fi. Dále se zabývá možnostmi vizualizace dat. Výsledkem této práce je vytvoření komunikační brány k IQMESH síti a mobilní aplikace, která s touto komunikační bránou komunikuje. Dále komunikační brána sbírá data z IQMESH sítě, ukládá je a posílá do mobilní aplikace k dalšímu zpracování.

Klíčová slova: IQRF, IQMESH, Bluetooth, Wi-Fi, Raspberry Pi, Android

Abstract

This thesis deals with the IQRF technology. It describes the Rasperry Pi 3 computer and its utilization within the IoT. The principles of Bluetooth and Wi-Fi communication technologies are discussed. Also the possibilities of data visualisation are covered. As the result of the thesis the communication gateway for the IQMESH network and mobile application communicating with the gateway were created. The communication gateway also collects data from the IQMESH network, stores them and sends them to the mobile application for further processing.

Key Words: IQRF, IQMESH, Bluetooth, Wi-Fi, Raspberry Pi, Android

Obsah

Seznam použitých zkratek a symbolů

DPA	–	Direct Peripheral Access
IoT	–	Internet of Things
FRC	–	Fast Response Command
MESH	–	Fast Response Command
RFCOMM	–	Radio Frequency Communication
HWP	–	Hardware Profile
IDE	–	Integrated Development Environment
STD	–	Standart
LP	–	Low Power
XLP	–	Extra Low Power
OS	–	Operační systém
SPI	–	Serial Peripheral Interface

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

V současné době je jedním z nejrozšířenějším pojmů "Internet věcí", neboli IoT. Je zde snaha připojit zařízení do společné sítě v rámci vzájemné komunikace. Často jsou využívány cloudy, které nabízí možnost ukládání dat přímo z IoT zařízení. Tyto cloudy lze dále použít pro komunikaci se zařízeními a pro vizualizaci uložených dat.

Druhá kapitola této práce se zabývá popisem počítače Raspberry Pi a technologie IQRF. Popisuje parametry počítače a jeho možnosti, které jsou důležité pro využití v rámci této práce. Dále kapitola obsahuje popis technologie IQRF. Zmiňuje komunikační schopnosti, využití v praktických řešeních a jednotlivé parametry této technologie. Třetí kapitola popisuje komunikační technologie Bluetooth a Wi-Fi. Obsahuje popis důležitých parametrů a komunikačních protokolů. Lze zde vyčíst rozdíly mezi těmito technologiemi. Další kapitola se zaobírá vizualizačními prostředky. Nejprve obsahuje popis nejpoužívanějších IoT cloudů a to IQRF Cloud, Microsoft Azure a IBM Watson IoT. Poté popisuje operační systém Android, který lze využít v rámci vizualizace. Pátá kapitola popisuje návrh a následné řešení komunikační brány. Obsahuje detailní popis zpracování jednotlivých částí, tzn. popis jednotlivých knihoven, algoritmů a implementaci programu do počítače Raspberry Pi 3. V další kapitole je popsán návrh a postup řešení prototypové mobilní aplikace, která bude navazovat na řešení komunikační brány. Předposlední kapitola shrnuje celkové řešení práce v rámci demonstračních měření. Zobrazuje funkčnost jednotlivých části řešení.

2 Raspberry Pi a technologie IQRF

2.1 Technologie IQRF

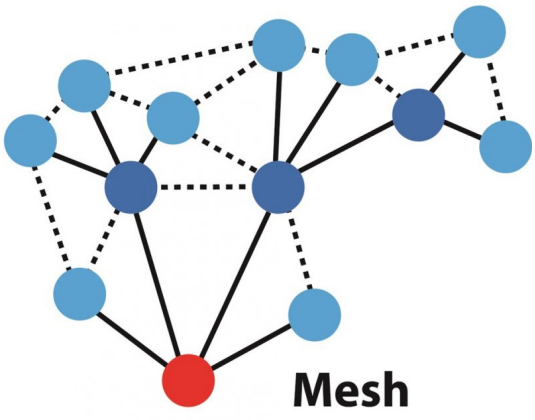
IQRF je bezdrátová nízko-příkonová komunikační technologie. Její využití je možné v mnoha IoT zařízeních, jak pro řízení v domácích podmínkách, tak i v průmyslových aplikacích pro sběr dat a řízení. Komunikace běží ve volných, bezlicenčních pásmech 433 MHz, 868 MHz a pro americký kontinent 916 MHz. V každém pásmu je možno využít několik kanálů, které mohou běžet současně. Technologie využívá komunikační protokol IQMESH, který používá typologii MESH sítě. Podobnou bezdrátovou komunikací je i ZigBee [1].

Vytvořit IQRF síť lze dvěma způsoby. Prvním způsobem je využití už dříve zmiňovaného protokolu IQMESH, který využívá typologii MESH s DPA dotazováním. Další možností je naprogramovat vlastní komunikační procesy.

2.1.1 Princip IQMESH sítě

Základem IQMESH sítě je koordinátor. Ten řídí provoz sítě a jeho běh probíhá synchronně. Dotazuje se ostatních zařízení, která se nazývají uzly. V takové síti je možno mít až 239 uzlů. Když je vyslán příkaz, koordinátor ho rozešle uzlům, na které vidí přímo, a ty ho rozešlou dál. To znamená, že uzly fungují zároveň i jako směrovače. Každý uzel opakuje signál ve svém časovém intervalu. Proto nedochází k vzájemným kolizím. Tak může dojít příkaz až k nejvzdálenějšímu zařízení. Vždy se doporučuje, aby každý uzel měl kolem sebe více než jednoho souseda, tedy další uzel, který by signál opakoval. Takto postavená síť je velice robustní a není náchylná k selhání kvůli výpadku některého z uzlů, nebo rušení, které by mohlo zastavit zasílání příkazu [2].

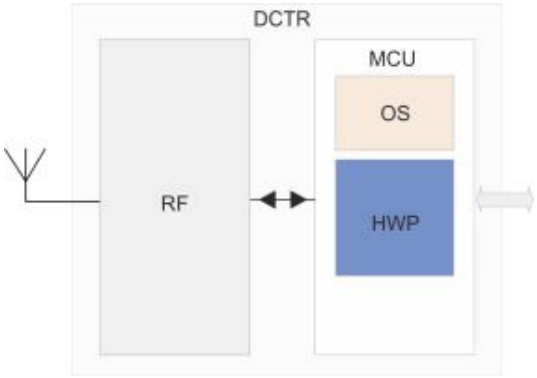
Ilustrace znázorňuje strukturu MESH sítě (Obr. 1). Červené zařízení představuje koordinátora. Modrá zařízení jsou uzly a tmavě modrá znázorňuje uzly, které fungují jako směrovače.



Obrázek 1: Struktura MESH sítě

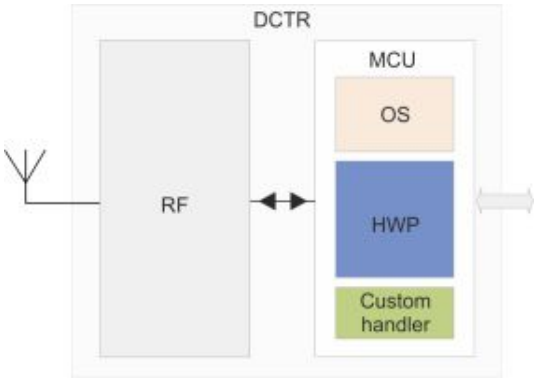
2.1.2 DPA dotazování

Samotné DPA dotazování lze rozdělit do tří vrstev (Obr. 2). První vrstvou je operační systém, na kterém DPA dotazování funguje. Další vrstvou je samotné DPA dotazování. To je připraveno k použití již bez jakéhokoliv kódového zásahu [2].



Obrázek 2: Blokové schéma modulu

Třetí vrstvou je rozšíření druhé vrstvy o další služby (Obr. 3). Tyto služby vytváří přímo uživatel, a to pomocí programovacího jazyka C. Toto rozšíření nazýváme Custom DPA handler. Ve většině aplikací si však uživatel vystačí se základními DPA dotazy a není potřeba funkce rozšiřovat [2].

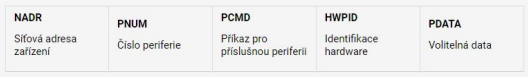


Obrázek 3: Blokové schéma modulu s rozšířením Custom DPA handler

2.1.3 Struktura DPA paketu

- Pole NADR (2B) – síťová adresa – může obsahovat hodnoty 0x00 (Koordinátor), 0x01 – EF (adresy uzlů), 0xFF (broadcast), 0xFE (dočasná adresa), 0xFC („local device“), ostatní adresy jsou rezervované.

- Pole PNUM (1B) – číslo periferie – např. 0x02 – OS, 0x03 – EEPROM, 0x04 – EEEPROM, 0x05 – RAM, 0x06 – LEDR, 0x07 – LEDG, 0x08 – SPI, 0x09 – IO, 0x0A – teploměr, 0x0D – FRC, 0x20–6F – uživatelské periferie.
- Pole PCMD (1B) obsahuje kód příkazu, který přísluší zvolené periferii. Může obsahovat hodnoty 0x00-0x3E.
- Pole HWPID (2B) slouží k filtrování cílového hardware. V případě, že je zde uvedena hodnota 0xFFFF, tak k filtrování nedochází.
- Pole PDATA může obsahovat až 56 bajtů uživatelských dat (v aktuální DPA verzi).



Obrázek 4: Struktura DPA paketu

2.1.4 FRC dotazování

Sbírání dat ze všech uzlů jednotlivě je velice časově náročné. Pro rychlejší získání dat je určeno FRC dotazování. FRC dotaz je určený pro koordinátora sítě. Ten se jednotlivých uzlů dotáže a přidá přijatá data na jejich předem určené místo do FRC odpovědi. Odpověď tedy tvoří jedna zpráva [1].

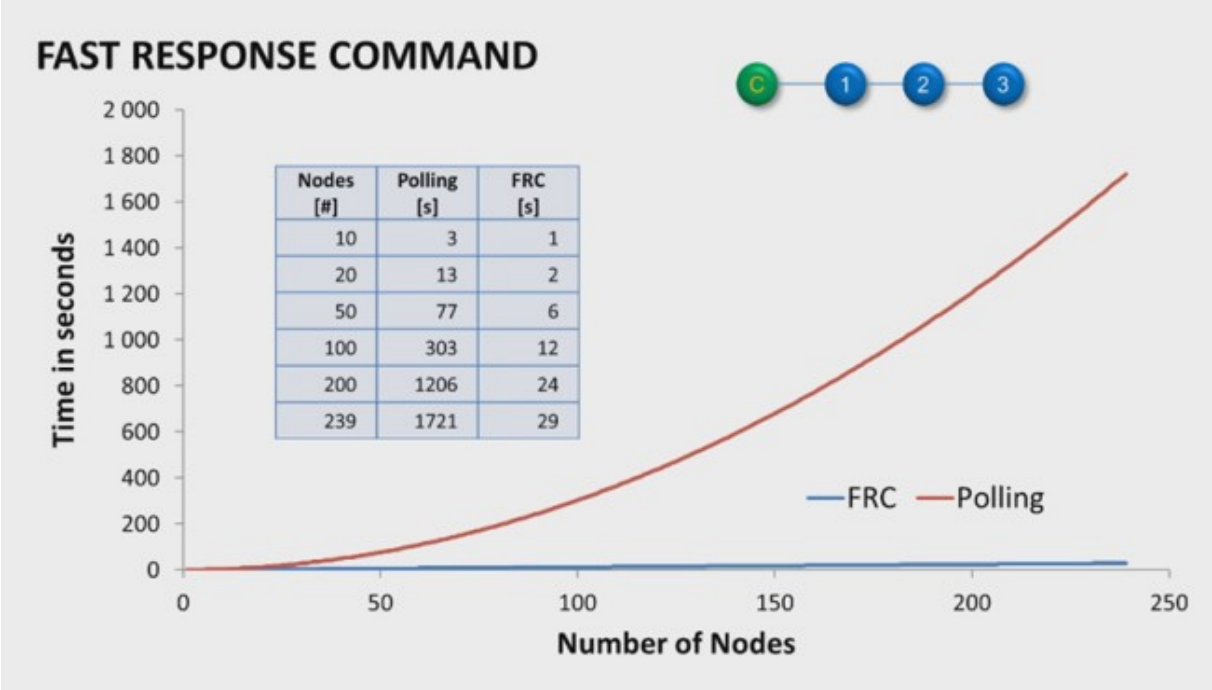
Jelikož je zpráva omezena pouze na 64 bytů, je samozřejmé, že nelze získat v jedné zprávě velká data od všech uzlů v IQMESH síti. Pokud chce uživatel data od 239 uzlů, což je maximální počet uzlů v IQMESH síti, může získat od každého z uzlů dvoubitovou informaci. Ze 62 uzlů získá 1 byte a ze 31 uzlů 2 byty. Proto je důležité rozhodnout, pro jaký účel se FRC dotazování hodí, a pro jaký ne [2].

FRC je jednou z periférií koordinátora. Pro použití musí být tato periférie na koordinátorovi povolena. ID periférie je PNUM = 0D. Poté musí být identifikován druh příkazu, a to nastavením parametru PCMD. Například PCMD = 00 je odeslání dotazu pro všechny uzly, PCMD = 02 je odeslání dotazu na vybrané uzly a PCMD = 01 je získání dostatečných výsledků příkazu FRC [4].

Na následujícím obrázku (Obr. 5) lze vidět rozdíl v rychlosti mezi klasickým dotazováním jednotlivých uzlů (polling) a FRC dotazováním [4].

2.1.5 Dosah

V MESH sítích je hlavním výhodou využívání uzlů jako směrovačů. Z toho plyne, že nezáleží na tom, jak daleko jsou ostatní uzly od koordinátora. Proto tyto sítě mohou dosahovat větších



Obrázek 5: Rozdíl rychlostí při dotazování na uzly v IQMESH síti

vzdáleností. Maximální vzdálenost mezi jednotlivými uzly ve vnitřních podmínkách, čili v budovách, se pohybuje okolo desítek metrů. Avšak ve venkovních podmínkách, kde se nenachází překážky, je maximální vzdálenost až 500 metrů. Toto platí pro tištěnou anténu. Připojením externí antény lze tuto vzdálenost zvětšit [1].

2.1.6 Napájení

Proudová spotřeba závisí zejména na režimu, ve kterém se zrovna modul nachází, a také na tom, jak dlouho v daném režimu modul setrvává. Proto je zde snaha zkrátit energeticky náročné procesy (režimy) na minimum [3].

Obecně je napájecí napětí IQRF modulu 3 V. Pro modul obsahující LDO je to od 3,1 do 5,3 V. Proudové odběry pak závisí na těchto režimech [3]. Sleep mód:

- 2,9 uA (všechny periferie včetně RF obvodu jsou vypnuty).

Run mód:

- RF sleep: 1,6 mA
- RF ready: 3,0 mA

RX mód:

- STD: 12,3 mA

- LP: 234 uA
- XLP: 16 uA

TX mód:

- 8,3 mA-19 mA (v závislosti na použitém vysílacím výkonu)

2.1.7 Specifikace přenosu dat

Transceivery IQRF umožňují přenášet data rychlostí 19,836 kb/s. Tato rychlost je pro řízení zařízení nebo přenos hodnot ze senzorů či stavů zařízení úplně dostačující. Není vhodná pro přenos velkých datových objemů – např. video nebo audio soubory [2].

Každý DPA paket umožňuje přenášet až 64 bytů uživatelských dat. Na množství těchto dat závisí délka vysílání daného paketu. Ta se pak pohybuje od 30 do 50 ms. Tento údaj je užitečné znát zejména pro výpočet doby, po kterou je síť obsazena vysíláním daného paketu. Pokud je např. v síti IQRF 100 směrujících uzlů a každý z nich vysílá paket 50 ms, trvá odvysílání zprávy celou sítí 5 s. Pokud se jedná o sběr dat, musí se počítat i s cestou zpět [1], [2].

2.1.8 Hardwarové vybavení

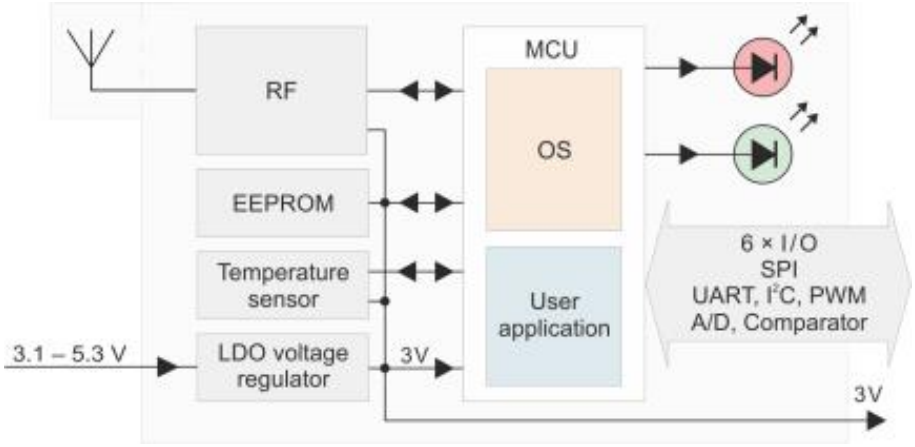
Základem celého systému jsou transceivery. Ty mohou být v různých velikostech podle typu použití a zapojení. Nejčastěji používaným typem je však typ TR-72D (Obr. 6), který používá připojení přes SIM konektor. Umožňuje jednoduché vložení a vyjmutí bez nutnosti pájení [3].

Obrázek 6: TR-72D

Technický popis:

- Operační systém
- DPA Framework
- Serial EEPROM 32kB
- GFSK modulace

- RF výstupní výkon 8 mW
- PWM
- Rozhraní SPI, I2C, UART
- 6x I/O
- Indikační LED diody (červená, zelená)
- Programovatelný HW Timer
- +3 V LDO regulátor
- A/D převodník (2 kanály), komparátor
- Teplotní senzor
- Anténa na DPS, nebo možnost připojení externí antény pomocí U.FL konektoru
- Frekvenční pásma:
  - 868 MHz a 916 MHz
  - 433 MHz jen ve zvláštních případech u typu TR-72Dx-433

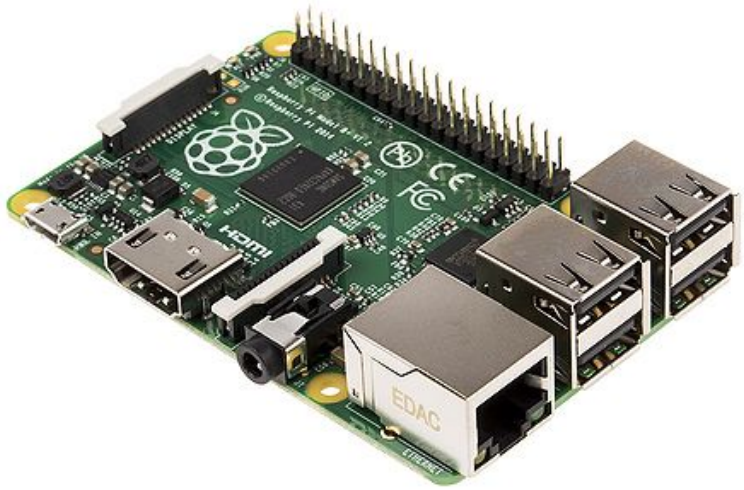


Obrázek 7: Blokové schéma zapojení

2.2 Raspberry Pi

Raspberry Pi je malý jednodeskový počítač. Byl vyvinut britskou nadací Raspberry Pi Foundation s cílem podpořit výuku informatiky na školách. Měl seznámit studenty s fungováním počítače a naučit je, jak ho použít pro řízení jiných zařízení. Jeho výhody spočívají hlavně ve

velikosti, která odpovídá přibližně velikostí platební karty, a také ve snadno přístupných periférií-ích, ve výkonu a ceně, která se pohybuje okolo 1200 Kč. Raspberry Pi také disponuje operačním systémem [11], [12], [13]. Všechny tyto parametry dělají z Raspberry Pi vhodný nástroj pro použití v oblasti IoT při sběru dat a následném zpracování. Proto je tento počítač použit i v této práci pro komunikaci s IQMESH sítí a zpracování měřených dat.



Obrázek 8: Raspberry Pi 3

2.2.1 Operační systém

Raspberry Pi je určen pro to, aby fungoval s operačním systémem. Ve většině případů se jedná o operační systém na bázi GNU/Linux. Systémy se od sebe liší podle míry optimalizace pro různé architektury procesoru. Níže jsou vypsány operační systémy spolu s popisem jejich vlastností, výhod a nevýhod. Všechny tyto operační systémy jsou nabízeny ke stažení na webových stránkách distributora Raspberry Pi [11], [12].

NOOBS (New Out of Box Software) – nejedná se o skutečný operační systém, ale o zaváděcí program zajišťující snadnou instalaci různých OS a jejich vyzkoušení. Uživatel dostane na výběr ze šesti OS a během každého spuštění Raspberry Pi si může vybrat jiný operační systém [11].

Raspbian je nejpoužívanější a doporučovaný nadací Raspberry Pi Foundation. Z tohoto důvodu stojí za Raspbianem obrovské množství uživatelů, kteří již vyřešili na veřejných fórech mnoho problémů, na které může uživatel při používání tohoto OS narazit. Podobně jako Debian je prezentován jako stabilní systém vhodný pro dlouhodobý provoz. Z těchto důvodů byl tento operační systém vybrán pro řešení této práce [11].

OpenELEC – XBMC Mediacenter distribuce je zjednodušený OS vhodný pro použití Raspberry Pi jako multimediálního centra. K přehrávání video souborů v kompresi MPEG-2 nebo VC-1 je nutno zakoupit kodeky, které jsou vázány na sériové číslo Raspberry Pi, a proto nejsou přenositelné [11].

RISC OS – GUI prostředí optimalizované pro běh při rozlišení 1080p. Je kompilováno pro architekturu ARM týmem lidí, kteří původně navrhli ARMové procesory. Nejedná se o derivaci Linuxu [11].

Arch Linux – Distribuce Arch Linux zkompilovaná pro ARM zařízení, je vhodná pro zkušenější uživatele, kterým umožňuje přizpůsobit si systém bez nadbytečných součástí[11].

3 Bezdrátové komunikační technlogie Bluetooth a Wi-Fi

3.1 Bluetooth

Bluetooth je bezdrátová komunikační technologie s krátkým dosahem a nízkou spotřebou energie. Může se použít jak pro přenos dat, tak pro přenos zvuku. Komunikace může probíhat mezi dvěma i více zařízeními. Tato technologie využívá frekvenční pásmo ISM, což je nelicencované pásmo [14], [15].

Bluetooth je definován standardem IEEE 802.15.1 a řadí se do skupiny počítačových sítí PAN. Technologie má již několik verzí, z nichž poslední je verze Bluetooth 5. Postupně se s novými verzemi objevuje snaha o co největší snížení spotřeby Bluetooth zařízení, což je důležitý faktor při návrhu IoT zařízení. Dále je zde samozřejmě snaha zvýšit dosah přenosu. Ten by měl od verze 4.0 v ideálních podmínkách dosahovat až 65 m [14], [15].

Tabulka 1: Rozdělení podle výkonosti

Class	mW	dBm	Dosah
1	100	20	100
2	2.5	4	10
3	1	0	1
4	0.5	-3	0.5

Tabulka 2: Přehled rychlostí podle verze

Bluetooth verze	Maximální rychlost	Maximální dosah
3.0	25 Mbit/s	10 metrů
4.0	25 Mbit/s	60 metrů
5.0	50 Mbit/s	240 metrů

V Bluetooth síti jsou zařízení identifikována pomocí své adresy BD_ADDR (BlueTooth Device Address) podobně, jako je MAC adresa v síti Ethernet. Síť může být dvoubodová nebo vícebodová. Více stanic může být připojeno do tzv. ad hoc sítě (piconet). Tam jedna stanice zastává pozici mastera a řídí komunikaci. Může obsloužit až 7 zařízení (slavů) [14].

Celá komunikace je řízená taktem řídicí stanice (mastera) způsobem přeskakování vysílacích kmitočtů. Do jednoho prostoru ve vzdálenosti 10 m se vejde až 10 picosítí. Ty se dále sdružují do rozprostřených sítí [14].

Bluetooth i Wi-Fi mají podobný princip ad-hoc komunikace. Wi-Fi oproti Bluetooth pracuje pouze na linkové vrstvě ISO/OSI modelu a nestará se o to, jaký typ dat přenáší. Bluetooth pracuje až s aplikační vrstvou modelu. Z toho vyplývá, že pro každý typ připojitelného zařízení musí mít Bluetooth speciální protokol. Až pak je s ním schopen komunikovat. To způsobuje

problém při vývoji aplikací, kde se musí hledět na kompatibilitu zařízení. Často tak vznikají chyby, kdy komunikace nefunguje [15], [14].

3.1.1 Podpora v operačním systému Linux

Pro operační systém Linux existují dvě implementace: BlueZ a Affix. BlueZ byl vyvinut společností Qualcomm a je součástí většiny linuxových jader. Knihovna BlueZ je proto použita i v této práci pro komunikaci pomocí technologie Bluetooth [7].

3.1.2 Protokol RFCOMM

Protokol RFCOMM je zkratkou Radio Frequency Communication. Bluetooth protokol RFCOMM je jednoduchou sadou přenosových protokolů, založených na L2CAP protokolech, která poskytuje emulovaný RS-232 sériový port (až šedesát současných připojení Bluetooth přístrojů). Protokol je založen na ETSI standardu TS 07.10. RFCOMM je někdy nazýván Serial Port Emulation. Bluetooth sériové porty jsou založeny na tomto protokolu [14].

3.2 Wi-Fi

Zkratkou Wi-Fi se označuje několik komunikačních standardů IEEE 802.11. Ty popisují počítačovou bezdrátovou síť, kde se používají také názvy Wirelles LAN, nebo zkráceně WLAN. Wi-Fi využívá, stejně jako Bluetooth, bezlicenční pásmo ISM [16].

Původním záměrem při vývoji Wi-Fi bylo zajistit bezdrátovou komunikaci zařízení (počítačů) na lokální úrovni, tedy v síti LAN. Až později začaly být Wi-Fi sítě připojovány do Internetu pomocí tzv. hotspotů. Dnes má technologii Wi-Fi zabudovanou většina zařízení: mobilní telefony, notebooky, chytré hodinky atd. Tak rozsáhlé využívání bezlicenčního pásma má však negativní důsledky. Jedná se především o zahlcení frekvenčního spektra a větší prolamování bezpečnosti. Novou generací počítačové bezdrátové technologie by měla být technologie WiMAX, která by kromě větší bezpečnosti měla zajišťovat i větší dosah připojení [16], [17].

Jak už bylo zmíněno v podkapitole o Bluetoooth, technologie Wi-Fi pracuje pouze na linkové a fyzické vrstvě. Tudíž se nestará o typ dat, která přenáší. Zde je pro přenos (šíření elektromagnetického pole prostorem) používán protokol CSMA/CA [16].

Základem navázání Wi-Fi spojení je identifikátor SSID (Service Set Identifier). SSID je řetězec až 32 ASCII znaků. Těmito identifikátory se Wi-Fi sítě od sebe odlišují. Ve většině případů je SSID vysílán všesměrově, tedy broadcast, čímž se síť ukazuje zařízením, která se k síti mohou připojit. Pokud má být síť skrytá, SSID se jednoduše přestane vysílat. Pokud se pak někdo bude chtít do sítě připojit, bude muset SSID znát [16], [17].

Aby mohla mezi sebou komunikovat zařízení od různých výrobců, jsou zde mezinárodní standardy. Těmi se zabývá institut IEEE (Institute of Electrical and Electronic Engineers). Standardy pro bezdrátové sítě jsou uváděny pod číslem 802.11. To pak lze dělit podle revizí.

Nejčastěji používané jsou 802.11b a 802.11g. Tyto revize se liší maximální rychlostí. U 802.11b je rychlost 11Mb/s, u 802.11g až 54Mb/s [16].

3.2.1 TCP

TCP je jedním ze soustavy protokolů TCP/IP. Je to protokol transportní vrstvy a zajišťuje spolehlivý přenos. To znamená, že příchozí zprávy jsou potvrzovány. Tím se liší od protokolu UDP. Protokol TCP na straně odesílatele postupně akumuluje jednotlivé byty do vhodné vyrovnávací paměti (bufferu) a po jejím naplnění odešle celý její obsah najednou v tzv. segmentu. Na straně příjemce se datový obsah segmentu analogicky ukládá do vyrovnávací paměti a jednotlivé byty jsou entitám aplikační vrstvy poskytovány z této vyrovnávací paměti [9].

4 Vizualizace

Tato kapitola nejprve shrnuje základní možnosti ukládání dat. Nejčastěji se to děje za pomoci cloudů, které jsou nejběžnějším prostředkem pro uložení měřených dat. Cloudy nabízí několik možností zpracování dat a vizualizace a také možnost zasílání příkazů IoT zařízením.

V této práci však cloudy nejsou využity a pro zpracování dat a vizualizaci je použit systém Android. Proto tato kapitola obsahuje i seznámení s tímto systémem.

4.1 IQRF Cloud

IQRF Cloud je výchozím cloudem pro veškerá IQRF zařízení. Je produktem firmy CIS, s.r.o., která patří do IQRF Alliance. Jeho největší výhodou je možnost používání zdarma. Tím se liší od jiných cloudů. Nenabízí sice tak velké možnosti jako konkurence, ale jeho omezená funkcionalita nabízí postačující minimum. IQRF Cloud umožňuje registraci a připojení více zařízení. Ta je rychlá a uživatelsky přívětivá. Služba podporuje tvorbu maker a skriptů pro exportování načtených dat. Nabízí i cyklické zadávání příkazů [1].

4.2 Microsoft Azure

Azure je cloudové úložiště od firmy Microsoft. Je známé řadou dodržovaných předpisů, uznáním jako nejdůvěryhodnější cloud pro americké orgány státní správy a autorizací FedRAMP. Nabízí možnost hybridního užití - spojení cloudových služeb a aplikací pro PC v úzkém propojení. Azure uvádí, že má největší množství datových center po celém světě, navíc podporuje open-source projekty a je partnerem Red Hat, což umožňuje využití tohoto řešení pro různé platformy a vývojářské systémy. Zabezpečení je řešeno podobně jako u Amazon AWS pomocí identit jednotlivých zařízení, navíc má možnost využití nástroje pro hromadnou registraci zařízení, umělé inteligence a další pokročilé analýzy. Cena za využívání základní služby Microsoft Azure IoT Centrum se liší v závislosti na zvoleném tarifu, který se odvíjí od množství zpráv za den [18].

4.3 IBM Watson IoT

IBM Watson IoT je platforma pro IBM Bluemix. Představuje komplexní platformu, která obsahuje nástroje pro snadnou tvorbu a implementaci řešení IoT. Platforma Watson IoT je implementována jako služba v Bluemixu a je vystavěna nad následujícími klíčovými oblastmi [5].

- Watson IoT Platform Connect umožňuje připojení koncových zařízení od jednoduchých čipů, přes inteligentní spotřebiče až po aplikace a průmyslová řešení. Dále umožňuje správu koncových zařízení, škálování pomocí cloudových služeb a napojení na analytickou část.
- Watson IoT Platform Information Management umožňuje transformovat a ukládat data IoT. Umožňuje konzumaci dat z různých datových zdrojů a platforem a jejich analýzu.

- Watson IoT Platform Analytics umožňuje zobrazení a analýzu dat v reálném čase. Umožňuje také použití nástrojů pro kognitivní analytiku nad strukturovanými i nestrukturovanými daty.
- Watson IoT Platform Risk Management umožňuje správu rizika prostřednictvím přehledných dashboardů a pokročilých alertů. Umožňuje správu notifikací a incidentů z jedné konzole.

4.4 Operační systém Android

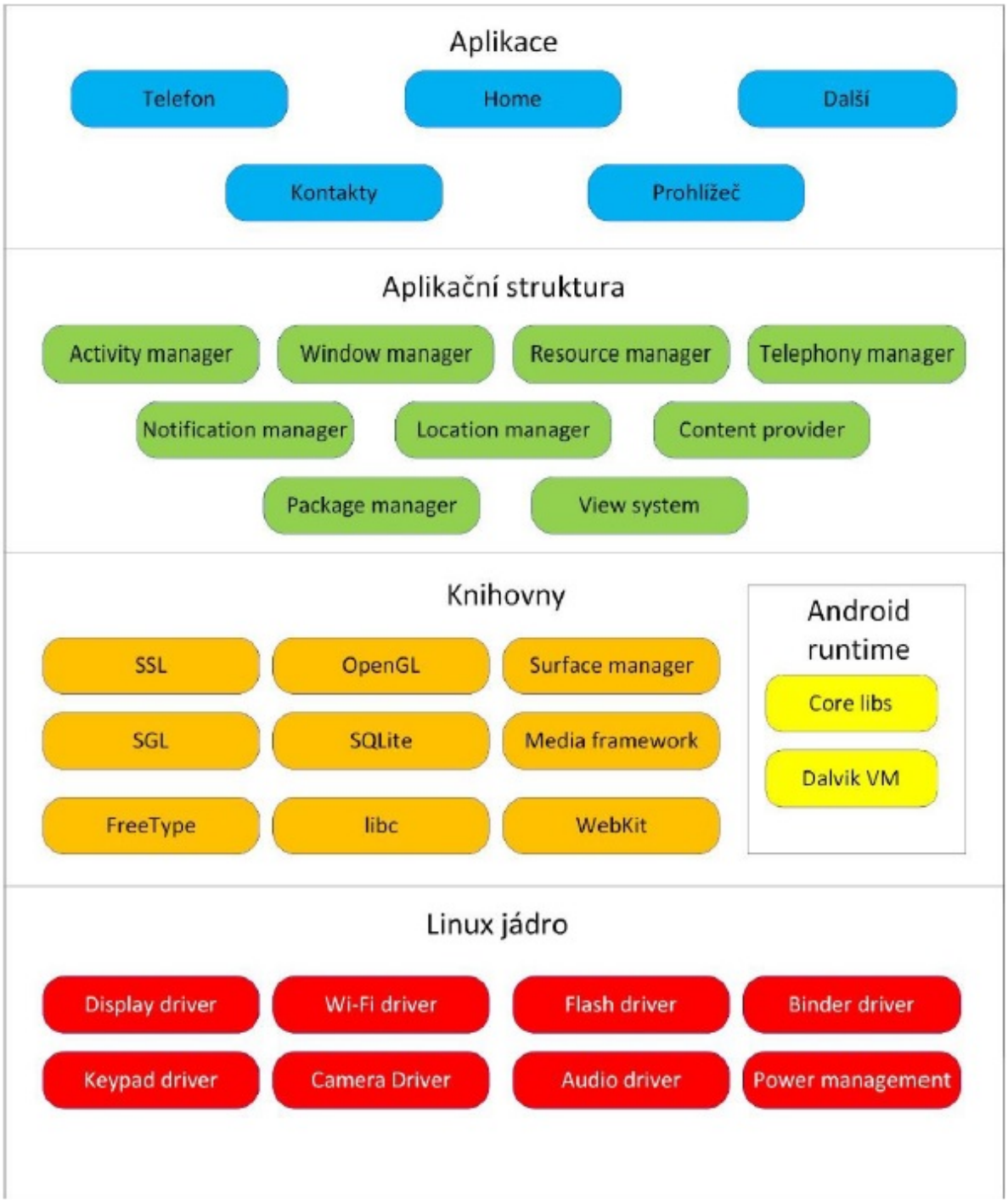
4.4.1 Základní informace

Android je komplexní open source platforma určená pro mobilní zařízení. Prosazuje ji společnost Google a je ve vlastnictví Open Handset Alliance. Android je první skutečně otevřenou platformou, která odděluje hardware od softwaru. To umožňuje mnohem větší počet zařízení, na kterých se dají spouštět stejné aplikace, a vzniká tak mnohem bohatší prostor pro vývojáře a spotřebitele. Pro vývojáře poskytuje všechny nástroje a frameworky pro vývoj mobilních aplikací.

4.4.2 Architektura OS

Operační systém Android se skládá z několika vrstev. Každá vrstva má své vlastní charakteristiky a účel, ale vrstvy nejsou vždy zcela oddělené a často se prolínají. Vrstvy OS:

- Linuxové jádro
- Knihovny
- Android Runtime
- Aplikační struktura
- Aplikace



Obrázek 9: Struktura Androidu

5 Komunikační brána na platformě Raspberry Pi 3

5.1 Návrh komunikační brány

Nejprve je důležité shrnout požadavky na komunikační bránu. Brána bude plnit funkci přeposílání DPA zpráv mezi IQMESH sítí a mobilní aplikací. Pro komunikaci s mobilní aplikací bude možno využít komunikačních technologií Bluetooth a Wi-Fi. Tyto komunikace by tedy měly být obě přístupné z vnějšku a uživatel, který by se chtěl k bráně připojit, by měl mít možnost si komunikaci vybrat.

Z tohoto návrhu tedy plyne, že obě komunikace musí být najednou v režimu, ve kterém čekají na spojení. To by mělo být zajištěno použitím dvou programových vláken. Ta by měla být na sobě nezávislá do doby, kdy se jedné komunikaci podaří uskutečnit spojení. V té chvíli by mělo být čekání na spojení druhé komunikace ukončeno. Nesmí dojít k tomu, aby se jiný uživatel připojil k jedné komunikaci poté, co bylo provedeno spojení druhou komunikací. To by způsobilo kolaps a program by se ukončil.

Po úspěšném spojení jedné z komunikací by tedy program měl přejít do režimu brány, kde bude čekat na přijetí DPA zprávy ze strany mobilní aplikace a zároveň ze strany IQMESH sítě. Mělo by zde být zajištěno asynchronní čtení zpráv, které nebude závislé například na tom, zda byl zaslán dotaz ze strany mobilní aplikace.

Komunikační brána bude dále obsahovat možnost sběru dat z IQMESH sítě, tedy logování. To by mělo fungovat zcela nezávisle na průběhu samotné brány. Z toho plyne, že proces sběru dat by měl fungovat v jiném vlákně. Logování bude muset zasílat a číst zprávy z IQMESH sítě, stejně jako komunikační brána. Proto je nutno zajistit, aby nedošlo ke konfliktu při přistupování ke komunikaci s IQMESH sítí.

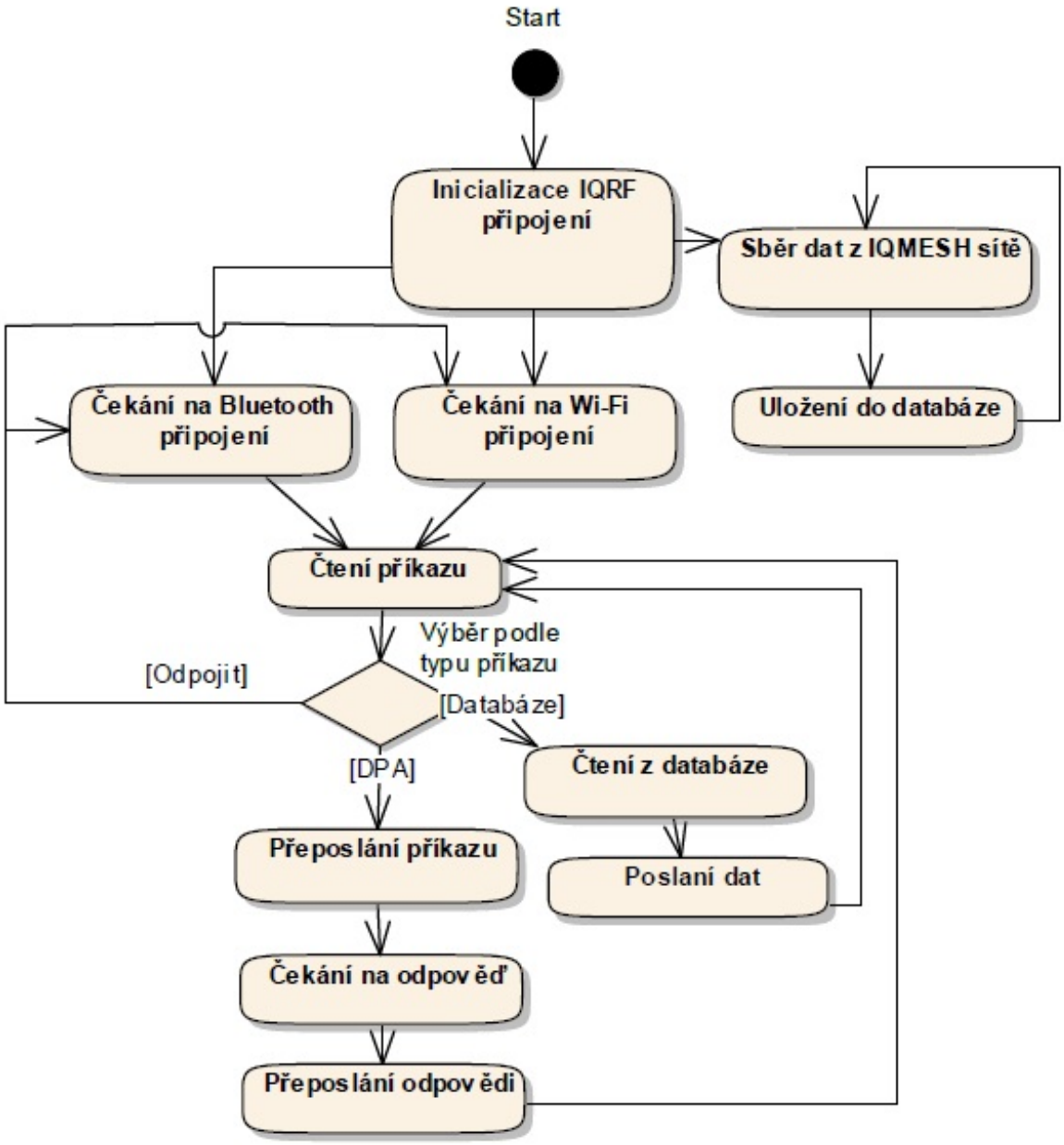
Logování je proces, v němž je potřeba vyčítaná data ukládat do paměti. V případě této práce se jedná o paměť SD karty, která je součástí počítače Raspberry Pi 3. Je tedy nutno zajistit vhodný způsob uložení dat a vytvořit tak lokální databázi. Logovaná data z jednotlivých uzlů je potřeba vhodně uložit ve formě, kdy pro jeden uzel bude načítána jedna hodnota.

Poté je zapotřebí zajistit čtení dat z databáze, která obsahuje nasbírané záznamy. Proto bude potřeba rozšířit komunikaci mezi mobilní aplikací a komunikační bránou o určitou formu příkazů a odpovědí. Těmi by mělo být zajištěno čtení z databáze podle určitých kritérií. Z toho plyne, že bude potřeba rozpoznávat typ zprávy - zda se jedná o DPA dotaz nebo o obslužný příkaz.

Tyto příkazy bude vhodné využít i při správě logování. Uživateli mobilní aplikace tak bude umožněno zaslat příkaz pro spuštění nebo zastavení logování, nebo nastavení logovacího intervalu.

Základní popis je shrnut v následujícím stavovém diagramu.

.



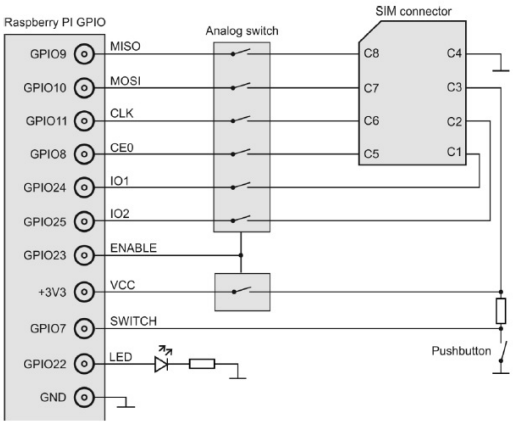
Obrázek 10: Stavový diagram IQRF serveru

5.2 Připojení IQRF k Raspberry Pi 3

Pro hardwarové připojení IQRF k Raspberry Pi slouží adaptér KON-RASP-01. Adaptér má SIM konektor pro připojení IQRF modulu. V tomto případě to je koordinátor IQMESH sítě. Adaptér zprostředkovává spojení pomocí komunikace SPI.

Adaptér je určen pro moduly TR-72D nebo TR-52D. Napájen je z Raspberry Pi a to 3,3 V, viz Obr. 12. Pro aktivaci je zapotřebí nastavit pin *enable* na logickou 1. Ten spojí napájení s adaptérem.

Obrázek 11: KON-RASP-01



Obrázek 12: Schéma KON-RASP-01

5.3 IQRF komunikace

Při testování IQRF komunikace na Raspberry Pi se vycházelo z podpory firmy IQRF. Firma IQRF nabízí vývojářský balík (SDK), který zahrnuje programy a knihovny použitelné pro mnoho platforem, včetně Raspberry Pi. Při tvorbě řešení je možno z těchto programů a knihoven vycházet. Programy jsou vytvořeny jako demonstrační aplikace, jak pracovat s IQRF na Raspbery Pi.

Základem je schopnost poslat DPA paket do IQMESH sítě, přesněji koordinátorovi. Jak už bylo řečeno, pro propojení koordinátora s Raspberry Pi slouží adaptér KON-RASP-01. Ten komunikuje s Raspbbery Pi pomocí komunikace SPI. Použité SDK nabízí knihovnu *rpi_spi_iqrf*. Ta umožňuje jednoduchou obsluhu komunikace SPI bez nutnosti znát hlubší specifikaci.

Knihovna nabízí základní funkce pro práci s IQRF. Základem je inicializační funkce, která naváže spojení s koordinátorem. Vstupním parametrem je odkaz na použitý SPI adaptér. Dalšími funkcemi jsou funkce pro zápis a čtení. V obou případech je parametrem pole bytů a délka pole. V případě zápisu pole obsahuje data k zaslání, přesněji DPA zprávu. V případě čtení se jedná o ukazatel na pole, do kterého se uloží přijatá data. Poslední důležitou funkcí je ukončení spojení.

```
extern int rpi_spi_iqrf_init(const char* dev);
extern int rpi_spi_iqrf_initDefault(void);
```

```
extern int rpi_spi_iqrf_getSPIStatus(rpi_spi_iqrf_SPIStatus* spiStatus);
extern int rpi_spi_iqrf_write(void* dataToWrite, unsigned int dataLen);
extern int rpi_spi_iqrf_read(void* readBuffer, unsigned int dataLen);
extern int rpi_spi_iqrf_destroy(void);
extern errors_OperError* rpi_spi_iqrf_getLastError(void);
```

Výpis 1: Seznam všech funkcí v knihovně rpi_spi_iqrf

Funkce z knihovny *rpi_spi_iqrf* jsou zahrnuty v knihovně *iqrf*. Zde funkce obsahují všechna nastavení.

```
void printDataInHex(unsigned char* data, unsigned int length);
int InitIqrfConnect();
int ReadDpa(void* aMessage);
void DestroyIqrfConnect();
int Send(void *aMessage, int aLength);
int ReadFrc(void *aMessage);
int Read(void *aMessage);
```

Výpis 2: Definice funkcí vytvořené Bluetooth knihovny

Pro čtení odpovědí pro režim brány, slouží funkce *Read*. Ta nezahrnuje čekání na odpověď, ale pouze zkontroluje stav SPI linky a pokud není ve stavu READY, vrací 0. V opačném případě vrací délku zprávy a pomocí ukazatele i samotnou zprávu. Toto řešení bylo zapotřebí v případě, kdy je nutné číst, ale nebrzdit dlouho linku. Kontrola, zda byla obdržena odpověď z IQMESH sítě, se provádí v hlavním kódu a je možno ji kdykoli pozastavit. Tím se uvolní SPI pro jiné účely.

Opakem je funkce *ReadFrc*. Ta čeká na odpověď. Je určená pro režim logování, kde se má přečíst pouze první odpověď z IQMESH sítě.

5.4 Bluetooth a TCP komunikace

Tato podkapitola shrnuje použití jednotlivých komunikačních technologií v rámci použitého počítače Raspberry Pi 3. Kromě samotné obsluhy těchto periférií v kódu je zapotřebí i několik nastavení systému a instalací balíčků do systému Raspbian.

5.4.1 Bluetooth komunikace

Zde je použita knihovna BlueZ. BlueZ je nástroj pro práci s technologii Bluetooth na Linuxových systémech [7].

K instalaci BlueZ je zapotřebí několika příkazů.

```
sudo apt-get install bluetooth
```

```
sudo apt-get install bluez
sudo apt-get install blueman
```

Výpis 3: Instalce BlueZ

Základem je práce se socketem. Ten je nejprve zapotřebí alokovat. To se provede pomocí funkce *socket*, které se definuje rodina adres pro Bluetooth makrem *AF_BLUETOOTH*, typ socketu pomocí makra *SOCK_STREAM* a nakonec typ protokolu. V tomto případě to bude protokol RFCOMM. Proto se nastavuje makro *BTPROTO_RFCOMM*. Funkce vrací integer hodnotu.

```
int sock = socket(AF_BLUETOOTH, SOCK_STREAM, BTPROTO_RFCOMM);
```

Výpis 4: Alokování socketu

Pro nastavení lokálního adaptéru se musí nejprve inicializovat předdefinovaná struktura *sockaddr_rc*. Zde se nastavuje opět rodina adres *AF_BLUETOOTH*, typ Bluetooth adres *BDADDR_ANY* a číslo kanálu, ke kterému se socket bude připojovat.

```
int port = 1;
struct sockaddr_rc loc_addr = { 0 }

loc_addr.rc_family = AF_BLUETOOTH;
loc_addr.rc_bdaddr = *BDADDR_ANY;
loc_addr.rc_channel = (uint8_t) port;
```

Výpis 5: Definice adresování adaptéru

Poté se musí socket svázat s kanálem pomocí funkce *bind*.

```
int result = bind(sock, (struct sockaddr *)&loc_addr, sizeof(loc_addr));
```

Výpis 6: Svázání soscketu s kanálem

Než se nechá socket čekat na připojení, musí se registrovat služba. Ta se registruje pro konkrétní port a přidělí se jí identifikační klič UUID. Pro složitost je zde uvedena pouze odkazující funkce.

```
register_service(port);

result = listen(sock, 1);
```

Výpis 7: Svázání socketu s kanálem

Jakmile jiné zařízení chce navázat spojení, je připojení potvrzeno pomocí funkce *accept*.

```
opt = sizeof(rem_addr);
int client = accept(sock, (struct sockaddr *)&rem_addr, &opt);
```

Výpis 8: Potvrzení spojení

Nyní, když je navázáno spojení, je možno komunikovat. Pro komunikaci zde slouží jednoduché funkce *write* a *read*. Vstupem funkce *write* je identifikátor připojení, který vrátila funkce *accept*, a pak data, která se budou posílat. Ta představuje pole bytů a délka řetězce, tedy počet bytů. Funkce potvrdí odeslání vrácením počtu poslaných bytů, který by se měly shodovat s vloženým počtem.

```
int bytes_sent = write(client, message, sizeof(message));
```

Výpis 9: Zápis dat pomocí funkce write

Funkce *read* má vstupní parametry podobné. Avšak vložené pole zde představuje ukazatel, do kterého budou vložena obdržená data a počet bytů určuje maximální počet bytů, které se budou přijímat. Návratová hodnota je počet obdržených bytů. Funkce *read* je čekající funkce, podobně jako funkce *listen*.

```
bytes_read = read(client, input, 1024);
```

Výpis 10: Čtení dat pomocí funkce read

Použití těchto funkcí v hlavním kódu by nebylo vhodné kvůli přehlednosti. Většinu parametrů, které jsou zde použity, nebude potřeba v průběhu programu měnit. Také je zapotřebí zajistit sledování chyb, aby bylo zabráněno pádu programu. Proto je rozumnější vytvořit knihovnu, kde by byly tyto funkce implementovány do jednodušších funkcí a ulehčilo by se tak jejich použití.

Pro použití zde stačí 4 funkce. První se bude starat o inicializaci Bluetooth. To znamená nastavení parametrů, které byly zmíněny výše, čekaní na požadavek na připojení od jiného zařízení a navázaní spojení. Funkce vrací celočíselný identifikátor daného spojení.

Dalšími funkcemi jsou funkce pro zápis a čtení. Původní funkce *write* a *read* jsou sice samy o sobě jednoduché, avšak kód programu narůstá po použití jednotlivých opatření proti pádu programu.

Poslední funkcí je ukončení spojení. Zde se jen odkazuje na funkci *close*.

```
int InitServerBluetooth();
int ReadDataBluetooth(int client, char *input);
void WriteDataBluetooth(int client, void *message, int length);
void CloseServerBluetooth(int client);
```

Výpis 11: Definice funkcí vytvořené Bluetooth knihovny

5.4.2 Párování Bluetooth zařízení

Aby bylo možno navázat Bluetooth komunikaci mezi komunikační bránou, tedy Raspberry Pi 3, a mobilní zařízením, je nutno tato dvě zařízení spárovat. To lze provést pouze jednotlivými úkony na obou zařízeních.

Nejprve je potřeba v operačním systému Raspbian spustit správu Bluetooth příkazem **sudo bluetoothctl**. Zde se příkazem **scan on** spustí automatické vyhledávání dostupných zařízení. Poté je nutno na mobilním zařízení povolit Bluetooth viditelnost pro ostatní zařízení. V systému Raspbian se po vyhledání mobilního zařízení objeví název nalezeného zařízení a adresa.

Poté je nutno adresu použít pro párovaní, které se spustí příkazem **pair** spolu s adresou zařízení. Tím se spustí párování na obou zařízeních a je potřeba ho na obou stranách potvrdit.

Po úspěšném párování je nutno nastavit v systému Raspbian důvěru k mobilnímu zařízení. To se provádí příkazem **trust** spolu s adresou zařízení. Poté je možné, aby se mobilní zařízení mohlo kdykoliv připojit k Raspberry Pi 3 a mohlo komunikovat s komunikační bránou.

5.4.3 TCP komunikace

Postup inicializace TCP spojení je téměř identický jako u inicializace Bluetooth. Základem je opět socket. Zde se ve vstupních parametrech mění typ rodiny adres a to na *AF_INET*. Dále pak použitý komunikační protokol na *IPPROTO_IP*. Toto makro představuje TCP protokol [8].

```
int sock = socket(AF_INET, SOCK_STREAM, IPPROTO_IP);
```

Výpis 12: Alokování socket pro Wi-Fi spojení

Zde je definice adresování pro lokální adaptér.

```
int sock = socket(AF_INET, SOCK_STREAM, IPPROTO_IP);
```

Výpis 13: Definice adresování adaptéru

Postup při svázání socketu s kanálem, čekání na spojení a potvrzení spojení je taktéž stejný.

```
bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr))

listen(sockfd,5);

accept( sockfd, (struct sockaddr *) &cli_addr, (socklen_t*) &clilen) )
```

Výpis 14: Definice adresování adaptéru

Zápis a čtení dat pomocí funkcí *write* a *read*. Opět obě funkce vrací délku pole a hlavním vstupem je pole bytů.

```
int bytes_sent = write(client, message, sizeof(message));

bytes_read = read(client, input, 1024);
```

Výpis 15: Čtení a zápis

Stejně jako u Bluetooth je rozumnější vytvořit knihovnu, kde implementujeme jednotlivé funkce pro obsluhu TCP komunikace. Opět to bude nejprve funkce pro inicializaci TCP a navázaní spojení. Dále funkce pro zápis a čtení a nakonec funkce pro ukončení spojení.

```
int WriteDataTCP(int client, void *message, int length);
int ReadDataTCP( int client, char *input );
int InitServerTCP();
void CloseServerTCP(int client);
```

Výpis 16: Čtení a zápis

5.4.4 Vytvoření přístupového bodu Wi-Fi sítě

Aby bylo možné se ke komunikační bráně připojit pomocí TCP protokolu, musí být zajištěno, aby brána byla v jedné síti spolu se zařízením, které se chce k bráně připojit. Jednou z možností je připojit komunikační bránu spolu s mobilním zařízením do jedné sítě tvořené směrovačem (routerem), který by fungoval jako Wi-Fi přístupový bod. To však nelze zajistit v každém prostředí. Proto je výhodnější vytvořit Wi-Fi přístupový bod ze samotné komunikační brány. Uživatel mobilního zařízení se tak může připojit přímo ke komunikační bráně.

Nastavení přístupového bodu na Raspberry Pi 3 zahrnuje několik bodů. Nejprve je důležité aktualizovat systém příkazy **sudo apt-get update** a **sudo apt-get upgrade**. Poté je potřeba nainstalovat do systému Raspbian několik programů. K instalaci programu *hostapd* se použije příkaz **sudo apt-get install hostapd**. Pro instalaci programu *dnsmasq* se použije příkaz **sudo apt-get install dnsmasq** [10].

Po instalaci se programy automaticky zapnou. K jejich nastavení je však potřeba, aby byly nečinné. To je provedeno pomocí příkazů **sudo systemctl stop hostapd** a **sudo systemctl stop dnsmasq** [10].

Poté je potřeba otevřít v textovém editoru soubor **hostapd.conf**. To se provede příkazem **sudo nano /etc/hostapd/hostapd.conf**. Po otevření se do souboru nakopíruje následující nastavení a soubor se uloží. Těmito parametry se nastavuje například název sítě, tedy SSID. V tomto případě je SSID **Raspberry Pi**. Dále se nastavuje i heslo, a to **raspberry** [10].

```
interface=wlan0
driver=nl80211
```

```
ssid=Raspberry Pi
hw_mode=g
channel=6
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=raspberry
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Po uložení se musí otevřít soubor **hostapd** příkazem **sudo nano /etc/default/hostapd**. V souboru se musí změnit řádek *#DAEMON_CONF=""* na *DAEMON_CONF ="/etc/hostapd/hostapd.conf"*. Obdobně se řádek *DAEMON_OPTS=""* změní do podoby *#DAEMON_OPTS=""* [10].

Poté se příkazem **sudo nano /etc/dnsmasq.conf** otevře soubor **dnsmasq.conf**. Do něj se vloží následující nastavení [10].

```
#RPiHotspot config - No Intenet
interface=wlan0
domain-needed
bogus-priv
dhcp-range=192.168.4.100,192.168.4.200,255.255.255.0,12h
```

Položka *dhcp-range* nastavuje rozmezí možných IP adres, které mohou být přiděleny uživatelům, a masku sítě [10].

Poté se vytvoří záloha souboru **interfaces** příkazem **sudo cp /etc/network/interfaces /etc/network/interfaces-backup** a příkazem **sudo nano /etc/network/interfaces** se otevře soubor **interfaces**. Jeho obsah se vymaže a soubor se uloží [10].

Nyní se přechází k nastavení DHCP protokolu. Příkazem **sudo nano /etc/dhcpcd.conf** se otevře soubor **dhcpcd.conf**. Do něj se vloží následující nastavení a soubor se uloží [10].

```
nohook wpa_supplicant
interface wlan0
static ip_address=192.168.4.10/24
static routers=192.168.4.1
```

Řádkem *static ip_address* se nastavuje statická IP adresa komunikační brány [10].

Příkazem **sudo nano /etc/sysctl.conf** se otevře soubor **sysctl.conf**. Zde je potřeba změnit řádek *#net.ipv4.ip_forward=1* do podoby *net.ipv4.ip_forward=1*. Poté se soubor uloží [10].

Nyní se vytvoří soubor příkazem **sudo nano /etc/iptables-hs**. Do souboru se vloží následující text a soubor se uloží [10].

```
#!/bin/bash
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j
    ACCEPT
iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

Poté se příkazem **sudo chmod +x /etc/iptables-hs** přiřadí oprávnění souboru **iptables-hs**. Dále se vytvoří soubor **hs-iptables.service** příkazem **sudo nano /etc/systemd/system/hs-iptables.service**. Do souboru se vloží následující nastavení a soubor se uloží [10].

```
[Unit]
Description=Activate IPtables for Hotspot
After=network-pre.target
Before=network-online.target

[Service]
Type=simple
ExecStart=/etc/iptables-hs

[Install]
WantedBy=multi-user.target
```

Poté je příkazem **sudo systemctl enable hs-iptables** aktivován *service* soubor **hs-iptables**, který bude spuštěn pří každém spuštění systému [10].

5.4.5 Jednotná komunikační knihovna

Funkce vytvořené pro Bluetooth komunikaci a TCP komunikaci jsou shrnuty v knihovně *connection*. Ta nabízí lepší přehlednost a po připojení není potřeba kontroly, která komunikace je právě používána.

Pro inicializaci slouží funkce *InitConnection*. Ta nastavuje parametry obou komunikacím. Pro připojení jsou zde dvě funkce *WaitForBluetoothConnection* a *WaitForTcpConnection*. Po úspěšném připojení jedné z komunikací se nastaví globální proměnná *connectionType*, která určuje typ připojení.

Pro zápis je určena funkce *WriteMessage*, jejímiž parametry jsou ukazatel na pole se zprávou a délka zprávy. Pro čtení slouží funkce *ReadMessage*, jejímž parametrem je ukazatel na pole, do kterého se přečtená zpráva zapisuje. Funkce vrací délku přečtené zprávy.

Ukončení spojení provádí funkce *CloseConnection*. Ta ukončí spojení právě používané komunikace.

```
int WriteMessage(void *aBuffer, int aLength);
int ReadMessage(void *aBuffer);
int InitConnection();
int WaitForBluetoothConnection();
int WaitForTcpConnection();
int CloseConnection();
```

Výpis 17: Knihovna connection

5.5 Logování

5.5.1 Sběr dat

Sběr dat je implementován v knihovně *logging*. Pro spuštění logování slouží funkce *StartLogging*. Ta spustí vlákno, které reprezentuje funkce *LoggingThread*. Pro zastavení logování slouží funkce *StopLogging*.

Vlákno *LoggingThread* čte v cyklu aktuální čas a po uplynutí nastaveného intervalu spustí logovací proces funkcí *LoggingProcess*. Ta v sobě implementuje dříve zmíněný postup dotazování se. Knihovna dále obsahuje několik funkcí pro synchronizaci s jinými procesy v programu a nastavení a čtení logovacího intervalu.

Sběr dat zahrnuje čtení měřených hodnot z celé IQMESH sítě. V tomto kroku je otázkou, jakým způsobem číst měřená data. Jedna z možností je dotazovat se každého uzlu zvlášť. Zde však nastává problém s rychlostí, která se úměrně zvyšuje s počtem uzlů, ze kterých se bude číst. Druhou možností je využití již dříve zmíněného FRC dotazování. To zajistí rychlé dotázaní všech uzlů.

Nejprve je nutno se koordinátora dotázat, které uzly jsou do sítě připojené. To lze provést DPA dotazem *Get Bonded Nodes* (Tab. 3). Tím je získána mapa připojených uzlů, která je přepočtena na pole proměnné bool.

Tabulka 3: Dotaz Get Bonded Notes

NADR	PNUM	PCMD	HWPID
0	0	2	FFFF

Aby se mohla ukládat měřená data ve správném formátu, je zapotřebí znát, o jaký typ dat se jedná. Přesněji řečeno, co konkrétní uzel měří. Zde je potřeba se nejprve dotázat všech uzlů na informaci o jejich hardwarovém profilu (HWPID). Tento identifikátor určuje, pro jaký typ aplikace je konkrétní uzel určen. Každý DPA dotaz na konkrétní uzel vrací ve své odpovědi HWPID dotazovaného uzlu. V tomto případě byl použit dotaz Tab. 4

Tabulka 4: Dotaz na HWPID

NADR	PNUM	PCMD	HWPID
0	1	3F	FFFF

Pomocí funkce GetNodeTypeByHWPID z knihovny **database**, kde parametrem je přečtené HWPID, je získán typ dat. Tento proces je aplikován na všechny připojené uzly a je tak získán přehled o všech uzlech v síti.

Poté je možno přejít k samotnému poslání FRC dotazu. V tomto případě je použit dotaz (Tab. 6), který koresponduje s použitými Custom DPA handlery v uzlu. Zde je dotaz určen pro 62 uzlů. Po přijetí odpovědí je zpráva přečtena v cyklu po jednotlivých bytech a příslušné byty, uchovávající měřená data, jsou pomocí funkce *PrepareAndWrite* zpracovány.

Tabulka 5: FRC dotaz

NADR	PNUM	PCMD	HWPID	PDATA
0	D	0	FFFF	90.5E.1.0.0

Jelikož odpověď neobsahuje data od všech uzlů, je zapotřebí si příkazem *ExtraResults* (Tab. 6) vyžádat od koordinátora zbylá data. Na přijatou odpověď je aplikován stejný postup jako na předchozí odpověď.

Tabulka 6: dotaz Extra results

NADR	PNUM	PCMD	HWPID
0	D	1	FFFF

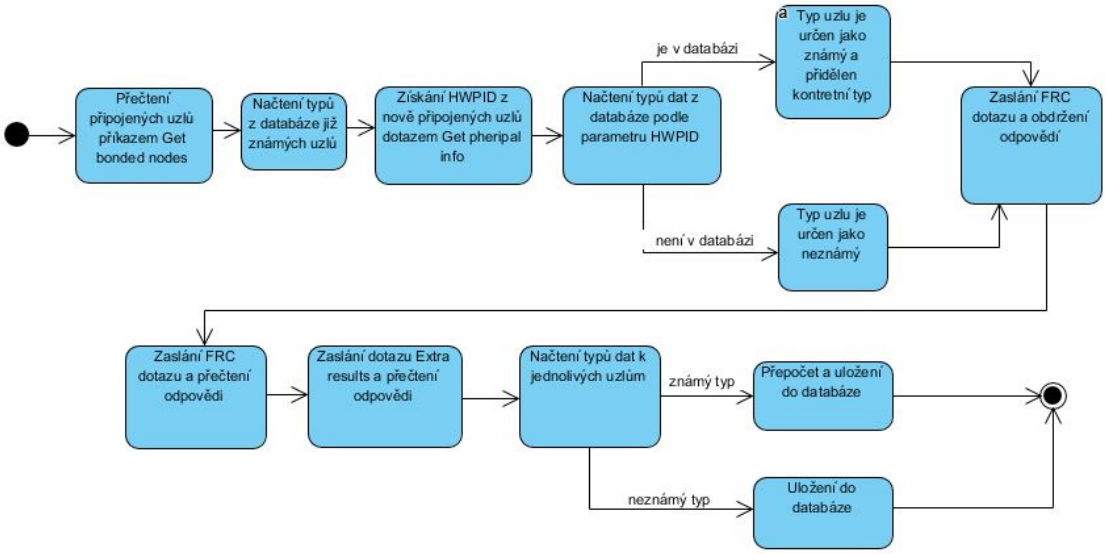
Řešení předpokládá, že uzel, který má být zahrnut v rámci sběru dat, obsahuje Custom DPA Handler. Dále předpokládá, že uživatelská periférie má definovaný hardwarový profil, určující, co konkrétní uzel měří.

Následující výpis zobrazuje funkce knihovny *logging*.

```
void *LoggingThread(void *arg);
int StartLogging();
int StopLogging();
int SetInterval(int aInterval_s);
int InitLogging();
int GetLoggingState();
int GetLoggingIsRunning();
int GetLoggingInterval();
```

Výpis 18: Knihovna logging

Celkový postup je zobrazený ve stavovém diagramu (Obr. 13).



Obrázek 13: Stavový diagram logování

5.5.2 Ukládání do databáze

Pro režim logování bylo zapotřebí využít databázi pro uložení čtených dat. Pro tuto práci byla použita databáze MySQL [6].

Instalace MySQL do operačního systému Raspbian zahrnuje tyto příkazy.

```
sudo apt-get install mysql-server
sudo apt-get install php5-mysql
```

Výpis 19: Instalace MySQL

Tato instalace však neobsahuje knihovny pro jazyk C. Ty je zapotřebí dodatečně nainstalovat. Instalace knihovny je přístupná pouze pro Raspbian verze Jessie. Proto bylo zapotřebí přepsat ručně soubor *sources.list* a změnit verzi Stretch na verzi Jessie. Po provedení příkazu *update* je možno knihovnu *libmysqlclient* nainstalovat. Po instalaci je samozřejmě nutno změny vrátit zpět [6].

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install libmysqlclient-dev
```

Výpis 20: Instalace knihovny libmysqlclient

Po úspěšné instalaci je k dispozici knihovna *mysql*. K obsluze databáze stačí sada následujících funkcí [6].

```
mysql_real_connect
mysql_close
mysql_query
mysql_stmt_init
mysql_stmt_prepare
mysql_stmt_bind_param
mysql_stmt_execute
mysql_stmt_close
```

Výpis 21: Funkce pro práci s databází

Pro jednoduchý přístup k databázi, zápisu a čtení byla vytvořena knihovna *database*. Ta obsahuje všechny potřebné funkce pro zápis v režimu logování a čtení pro využití v mobilní aplikaci.

```
int ConnectMysql();
int DisconnectMysql();
int AddNodeTable(int aNodeNumber);
int WriteLog(uint8_t aNodeNumber, int aValue, void* aUnit, int aUnitLength);
int ReadLog(int aNodeNumber, int aPositionFromLast, int* aId, int* aValue, void
    * aUnit, MYSQL_TIME *aTimestamp);
int CreateNodesListTable();
int RemoveNodeTable(int aNodeNumber);
int ReadByTime(int aNodeNumber, MYSQL_TIME aStartTime, MYSQL_TIME aStopTime,int
     aId[], int aValue[], MYSQL_TIME aTimestamp[]);
int GetCountOfReadByTime(int aNodeNumber, MYSQL_TIME aStartTime, MYSQL_TIME
    aStopTime, int* aCount);
```

Výpis 22: Knihovna database

Pro připojení k databázi a následné odpojení slouží funkce *ConnectMysql* a *DisconnectMysql*. Pro vytvoření tabulky pro záznam měřených dat slouží funkce *AddNodeTable*. Jejím parametrem je číslo uzlu. Po jejím vykonání se v databázi vytvoří soubor s názvem *Node* a číslem uzlu. Samotná tabulka má 4 sloupce. Prvním je sloupec ID. Ten se automaticky inkrementuje při zápisu od čísla 1. Druhý sloupec Value je pro měřenou hodnotu. Sloupec Unit slouží pro záznam jednotky měřené veličiny. Pokud veličina není známá, je buňka v tomto sloupci prázdná. Poslední sloupec Time je typu Timestamp. Zde se automaticky zapisuje aktuální čas při zápisu.

K zápisu měřených dat slouží funkce *WriteLog*. Ta vychází z předchozí funkce a zapisuje do dříve vytvořené tabulky data. Jejími parametry jsou číslo uzlu, podle něhož se vybere příslušná tabulka, měřená hodnota a jednotka. Pokud je jednotka neznámá, její obsah je prázdný.

Číst data lze dvěma způsoby. První možností je jednoduché čtení, kdy se čte pouze jeden záznam. Funkce se nazývá *ReadLog* a jejími parametry jsou číslo uzlu, pozice od posledního záznamu a poté ukazatele, do kterých se zapisují čtená data. Těmi jsou: ID, měřená hodnota, jednotka a čas záznamu.

Druhou možností čtení je čtení s časovým rozmezím. K tomu slouží funkce *ReadByTime*. Parametry funkce jsou: číslo uzlu, první časová mez, druhá časová mez, počet přečtených záznamů a ukazatele na pole ID, pole měřených hodnot a pole časů záznamu.

K odstranění tabulky se záznamy určitého uzlu slouží funkce *RemoveNodeTable*. Jediným parametrem je číslo uzlu.

5.6 Zpracování příkazů

Pro zpracování obslužných příkazů bylo zapotřebí použít strukturu příkazů, podle které by se příkazy rozpoznávaly a rozdělovaly parametry příkazů. Nejprve bylo zapotřebí určit typ příkazu. V tomto případě to je nultý byte v přijatém řetězci dat. Jeden byte může nabýt až 256 možností. To je v tomto případě dostačující. Ostatní byty příkazu jsou určeny pro parametry příkazu. Jejich počet a délka se liší podle typu příkazu.

Pro každý příkaz existuje také odpověď či více odpovědí. Proto bylo zapotřebí určit i strukturu těchto zpráv. Nultý byte odpovědi určuje opět typ příkazu, ke kterému je odpověď určena. Byte na pozici 1 určuje, zda bylo zpracování příkazu úspěšné. Pokud byte nabývá jakékoli nenulové hodnoty, jedná se o chybový stav.

Tabulka 7: Struktura příkazu

Byte 0	Byte 1 - N
Číslo příkazu	Parametry příkazu

Tabulka 8: Struktura odpovědi

Byte 0	Byte 1	Byte 2 - N
Číslo příkazu	Chyba	Data

První sada příkazů zahrnuje práci s databází. Prvním příkazem je zde čtení jednoho záznamu. Tento příkaz zahrnuje funkci *ReadLog*. Parametry jsou tedy číslo uzlu a pořadí dotazovaného záznamu od posledního záznamu. Odpověď na tento příkaz je rozdělena na 4 částí, a to identifikační číslo záznamu (ID), hodnota, jednotka a časové razítko.

Dalším příkazem je příkaz *ReadByTime*. Tento příkaz je také určen pro čtení, ovšem jeho parametrem není pořadí záznamu od posledního, ale časové rozmezí. Je zřejmé, že odpovědí pak nebude jeden záznam, ale několik záznamů. Ty jsou vráceny po jednom záznamu na zasílaný paket.

Tabulka 9: Příkaz *ReadLog*

Byte 0	Byte 1	Byte 2 - 3
1	Číslo uzlu	Pořádí záznamu od posledního

Tabulka 10: Odpověď na příkaz *ReadLog*

Byte 0	Byte 1	Byte 2 - 3	Byte 4 - 5	Byte 6	Byte 7 - 13
1	Chyba	Id	Hodnota	Jednotka	Časové razítko

Tabulka 11: Příkaz *ReadByTime*

Byte 0	Byte 1	Byte 2 - 8	Byte 9 - 15
2	Číslo uzlu	Počáteční čas	Konečný čas

Tabulka 12: Odpověď na příkaz *ReadByTim*

Byte 0	Byte 1	Byte 2 - 3	Byte 4 - 5	Byte 6	Byte 7 - 13
2	Chyba	Id	Hodnota	Jednotka	Časové razítko

Časové formáty jsou v příkazech a odpovědích reprezentovány pomocí 7 bytů.

Tabulka 13: Časový formát

Byte 0 - 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Rok	Měsíc	Den	Hodina	Minuta	Sekunda

Pro vymazání tabulky se záznamy je určen příkaz *RemoveNodeTable*. Jeho parametrem je pouze číslo uzlu.

Tabulka 14: Příkaz RemoveNodeTable

Byte 0	Byte 1
3	Číslo uzlu

Tabulka 15: Odpověď na příkaz RemoveTable

Byte 0	Byte 1
3	Chyba

Následující sada příkazů slouží ke správě logování. K nastavení logovacího intervalu slouží příkaz *SetLoggingInterval*. Parametrem je čas v sekundách.

Tabulka 16: Příkaz SetLoggingInterval

Byte 0	Byte 1 - 2
21	Interval (s)

Tabulka 17: Odpověď na příkaz SetLoggingInterval

Byte 0	Byte 1
21	Chyba

Pro zjištění aktuálního logovacího intervalu slouží příkaz *GetLoggingInterval*. Nemá žádné parametry. Odpověď obsahuje dotazovaný interval v sekundách.

Tabulka 18: Příkaz GetLoggingInterval

Byte 0
22

Tabulka 19: Odpověď na příkaz GetLoggingInterval

Byte 0	Byte 1	Byte 2 - 3
22	Chyba	Interval (s)

Pro spuštění nebo vypnutí logování je určen příkaz *SetLoggingState*. Jeho parametrem je pouze jeden byte, který reprezentuje booleovské hodnoty 0 a 1 pro nastavení stavu logování.

Tabulka 20: Příkaz SetLoggingState

Byte 0	Byte 1
23	Stav

Tabulka 21: Odpověď na příkaz *SetLoggingState*

Byte 0	Byte 1
23	Chyba

Příkaz *GetLoggingStat* je určen pro zjištění stavu logování. Nemá žádné parametry. Příkaz obsahuje pouze číslo příkazu. Odpověď obsahuje booleovskou hodnotu určující stav logování.

Tabulka 22: Příkaz *GetLoggingState*

Byte 0
24

Tabulka 23: Odpověď na příkaz *GetLoggingState*

Byte 0	Byte 1	Byte 2
24	Chyba	Stav

5.7 Bluetooth/IQRF a Wi-Fi/IQRF brána

Funkce Bluetooth/IQRF brány a Wi-Fi/IQRF brány je implementována v hlavním souboru programu. Základem je čekaní programu na připojení od jiného zařízení. V našem případě to je čekání na požadavek na připojení přes technologii Bluetooth a čekaní na požadavek na připojení prostřednictvím TCP protokolu, který je vyšší vrstvou Wi-Fi technologie.

Program je v této části rozdělen na dvě vlákna, která jsou reprezentována funkcemi *BluetoothWait* a *TcpWait*.

```
void *BluetoothWait(void * arg)
void *TcpWait(void * arg)
```

Výpis 23: Čekání na připojení

Po úspěšném připojení jednoho vlákna je druhé vlákno deaktivováno a program pokračuje dál. Poté program přechází do funkce brány, kdy čeká na přijetí zprávy pomocí funkce *ReadMessage*. Po přijetí zprávy je přečten první byte z této zprávy. Tento byte určuje, o jakou zprávu se jedná. Pokud se byte rovná 127, jedná se o zprávu pro komunikační bránu. Ostatní byty pak obsahují samotný DPA dotaz.

Tabulka 24: DPA dotaz

Byte 0	Byte 1 - N
127	DPA

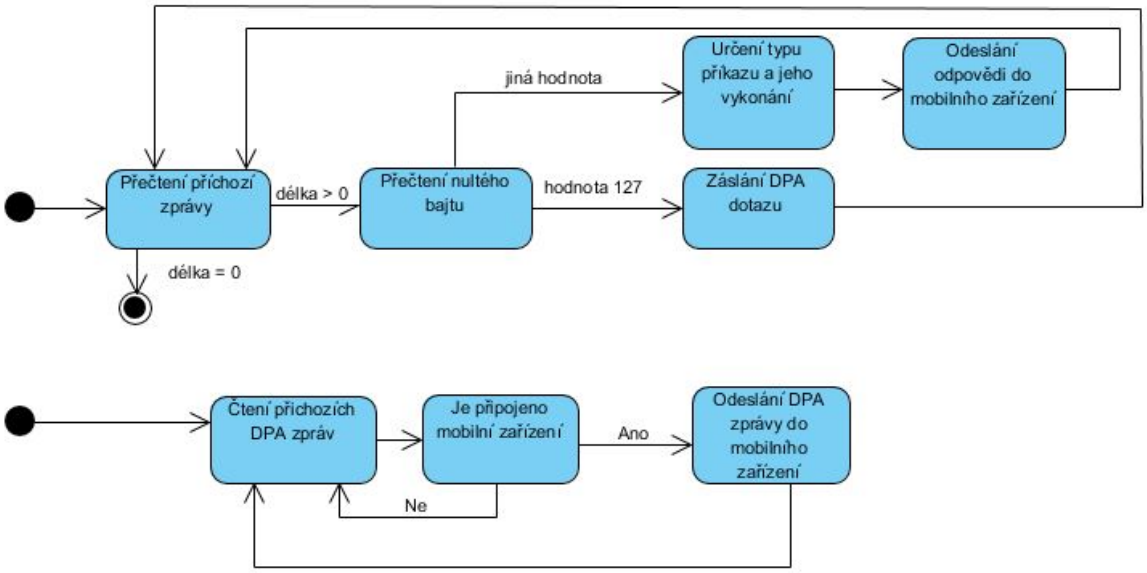
Zpráva je pomocí funkce *WriteDPA* zaslána do IQMESH sítě. Čtení odpovědí probíhá asynchronně ve vlákně reprezentovaném funkcí *ReadGateway*. Po obdržení odpovědí z IQMESH sítě je zpráva odeslána funkcí *WriteMessage*.

```
void *ReadGateway(void * arg)
```

Výpis 24: Vlákno pro čtení odpovědí IQMESH sítě

Pokud se první byte zprávy, vrácené funkcí *Read*, nerovná hodnotě 127, nejedná se o DPA dotaz, ale o obslužný příkaz. Pomocí funkce *DoCommand* je zpráva zpracována a do ukazatele na pole a délku v parametrech je zapsána odpověď. Ta je poté odeslána funkcí *WriteMessage*.

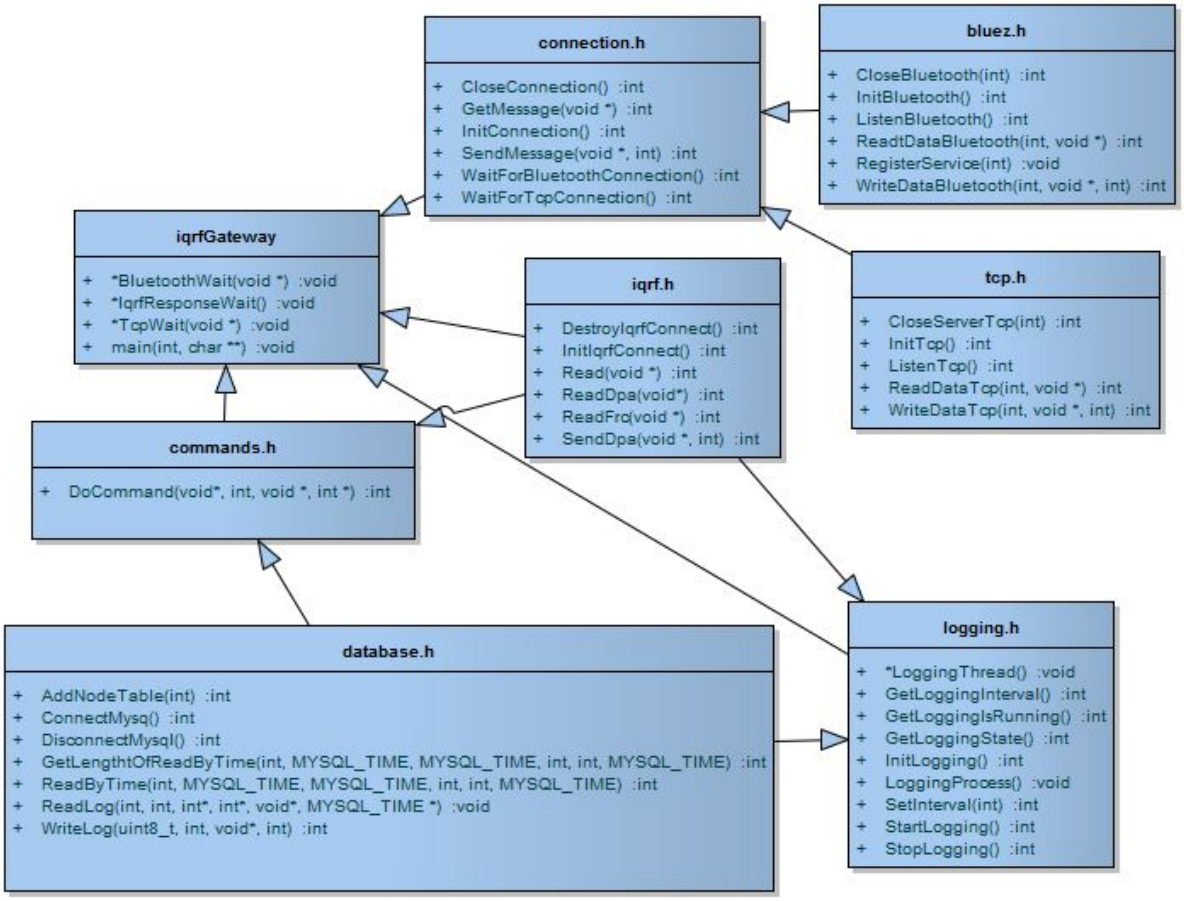
Tato sekvence je prováděna v cyklu, dokud není přijata zpráva na odpojení. Tato zpráva má nulovou délku. Celkový průběh je zobrazen ve stavovém digramu (Obr. 33).



Obrázek 14: Stavový diagram komunikační brány

5.8 Výsledná struktura komunikační brány

Na Obr. 15 lze vidět výslednou struktura komunikační brány. Struktura zobrazuje hlavičkové soubory programu a jejich funkce.



Obrázek 15: Struktura celkového systému

6 Prototypová aplikace pro sběr dat a správu bezdrátové sítě

6.1 Návrh mobilní aplikace

Výsledkem práce bude mobilní aplikace, která obsahuje grafické prostředí, v němž se bude pohybovat uživatel, který bude aplikaci využívat. Je tedy nezbytné se v návrhu aplikace zaměřit hlavně na toto grafické prostředí. K tomuto účelu nejlépe slouží diagram užití. V něm lze zaznamenat funkce a možnosti, které by měly být přístupné uživateli.

V mobilní aplikaci by první možností měl být výběr komunikační technologie, kterou bude chtít uživatel použít. To znamená vybrat, zda se ke komunikační bráně připojí pomocí technologie Bluetooth, nebo technologie Wi-Fi. Poté by uživatel aplikace měl mít možnost vidět seznam blízkých a dostupných zařízení, která využívají danou komunikační technologii. Po úspěšném připojení by uživatel měl být informován o úspěšném připojení. Než k tomu dojde, neměl by uživatel mít přístup k jiným částem aplikace.

Jakmile bude uživatel úspěšně připojen, bude si moci zvolit, jaký režim bude chtít využít. To bude vhodné vyřešit nabídkou služeb, režimů, ze kterých si uživatel vybere. Uživatel by měl mít možnost se mezi režimy volně pohybovat.

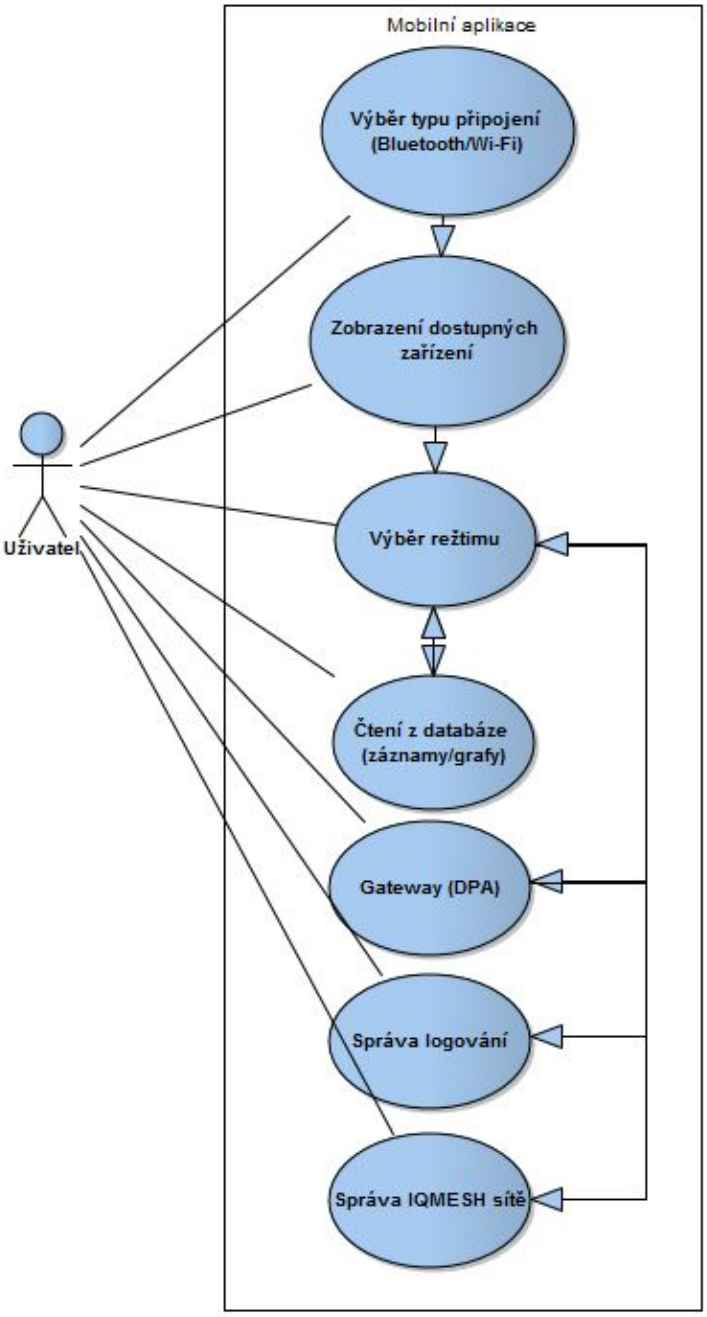
Prvním režimem bude zasílání DPA dotazů do komunikační brány. Zde by mělo být zajištěno jednoduché nastavení DPA dotazu, tedy nastavení jednotlivých částí zprávy. Měla by být zajištěna ochrana proti nastavení špatných parametrů dotazu. Při špatném nastavení dotazu by měl být uživatel vhodně informován. Po odeslání dotazu musí aplikace zachytávat odpovědi. Aplikace nebude vědět, zda opravdu odpověď obdrží, nebo ne. Proto by čtení z komunikační brány mělo fungovat asynchronně v jiném vlákně. To zajistí, že v případě, že nebude obdržena odpověď, aplikace na ni nebude čekat. Aplikace by měla po přijetí DPA odpovědi zprávu vhodně zobrazit. Zde lze vycházet z řešení v prostředí IQRF IDE. Je vhodné zobrazit taktéž typ zprávy.

Druhým režimem bude čtení nasbíraných dat z lokální databáze v komunikační bráně. Zde je nejprve potřeba zajistit nastavení vstupních parametrů, podle kterých se budou data číst. Opět by zde měla být ochrana proti špatným vstupním parametrům. Po odeslání příkazu by program měl čekat na odpověď. Zde je nutno zvážit, zda data posílat zvlášť po jednom záznamu, nebo v jedné zprávě. Jakmile budou všechny záznamy přijaty, měly by být ve vhodné formě zobrazeny. To znamená zobrazit měřenou hodnotu a čas, v němž byla hodnota zaznamenána.

Dalším režimem bude správa logování. Zde by měla být možnost logování spustit, nebo zastavit. Uživatel musí také vědět, v jakém stavu se právě logování nachází. Dále by měl uživatel mít možnost nastavovat logovací interval. Zde je samozřejmé, že by mělo být známo, jaký je aktuální interval.

Posledním režimem bude správa IQMESH sítě. To zahrnuje přidávání a odebírání uzlů ze sítě. Zde se dá vycházet z funkčního řešení z prostředí IQRF IDE, avšak stačí použít z něj jen hlavní dostačující funkce.

Možnosti užití mobilní aplikace jsou shrnuty v diagramu užití (Obr. 16).



Obrázek 16: Stavový diagram komunikační brány

6.2 Specifikace vybrané platformy a vývojového prostředí

Jako cílové zařízení byl vybrán mobilní telefon s operačním systémem Android 6.0 Marshmallow. Tato verze systému obsahuje API číslo 23. Obecně platí, že aplikace určená pro nižší verzi API je spustitelná na systémech s vyšší verzí API. Aplikace v této práci je cílená na verzi API 21. Proto je spustitelná na operačním systému Android Lollipop, který má verzi API 21-22.

Vybrané mobilní zařízení disponuje technologiemi Bluetooth a Wi-Fi. Ty jsou pro aplikaci, a celkově pro tuto práci, nezbytné.

Pro vývoj aplikace je vybraným prostředím Android Studio, konkrétně verze 3.1.1. Prostředí nabízí široké možnosti pro vývoj. Je uživatelsky přívětivé a je spojeno s velkou podporou ze strany vývojářů.

6.3 Bluetooth a TCP komunikace

6.3.1 Bluetooth komunikace

Pro obsluhu Bluetooth komunikace je zapotřebí naimportovat následující knihovny.

```
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
```

Výpis 25: Knihovny pro obsluhu Bluetooth

Nejprve se inicializuje Bluetooth adaptér na používaném zařízení. Do objektu třídy *BluetoothAdapter* se vloží používaný Bluetooth adaptér statickou metodou *getDefaultAdapter*.

```
bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
```

Výpis 26: Inicializace Bluetooth adaptéru

K definici zařízení, ke kterému se má připojit, slouží třída *BluetoothDevice*. Do instance této třídy se vloží vzdálené zařízení metodou *getRemoteDevice*, kde parametrem je SSID vzdáleného zařízení.

```
BluetoothDevice lBluetoothDevice = bluetoothAdapter.getRemoteDevice(aAddress);
```

Výpis 27: Určení vzdáleného zařízení

Samotná komunikace probíhá pouze pomocí objektu třídy *BluetoothSocket*. Socket se definuje metodou *createInsecureRfcommSocketToServiceRecord*, kde parametrem je UUID používané služby.

```
btSocket = lBluetoothDevice.createInsecureRfcommSocketToServiceRecord(myUUID);
```

Výpis 28: Definice UUID

Pro připojení Bluetooth komunikace slouží funkce *connect*.

```
btSocket.connect();
```

Výpis 29: Připojení ke vzdálenému zařízení

K zasílaní zpráv slouží funkce *write*. Argument funkce je pole bytů se zprávou.

```
btSocket.getOutputStream().write(lMessage);
```

Výpis 30: Zasílaná zpráv

Ke čtení příchozích zpráv je určena funkce *read*. Zde je parametrem ukazatel na pole bytů, do kterého se po přečtení zapíše příchozí zpráva a samotná funkce vrátí délku zprávy.

```
int lBytes = btSocket.getInputStream().read(lBuffer);
```

Výpis 31: Čtení přichozích zpráv

Ukončení spojení provádí metoda *close*.

```
btSocket.close();
```

Výpis 32: Ukončení spojení

Tyto metody byly implementovány do třídy *BluetoothThread*, která dědí vlastnosti třídy Thread. Tím se objekt této třídy stává dalším vláknem. To se po vytvoření instance spouští metodou *start*.

```
bluetoothThread = new BluetoothThread();
bluetoothThread.start();
```

Výpis 33: Spuštení vlákna pro Bluetooth komunikaci

6.3.2 TCP komunikace

Pro obsluhu TCP protokolu je zapotřebí naimportovat sadu knihoven.

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.InetAddress;
import java.net.Socket;
import java.io.IOException;
```

Výpis 34: Knihovny pro obsluhu TCP

Pro zasílání paketů je potřeba vytvořit objekt třídy *PrintWriter*. Pro čtení pak objekt třídy *BufferedReader*.

```
private PrintWriter BufferOut;

private BufferedReader BufferIn;
```

Výpis 35: Vytvoření objektů pro zápis a čtení

Tyto objekty poslouží jako zásobníky pro zápis a čtení.

K vytvoření spojení je potřeba znát IP adresu cílového zařízení. V tomto případě to je 192.168.4.10. Poté je potřeba znát i číslo portu používané služby, tedy 51717.

```
public static final String SERVER_IP = "192.168.4.10";
public static final int SERVER\_PORT = 51717;
```

Výpis 36: IP adresa a číslo portu

IP adresu je potřeba převést na formát *InetAdress*.

```
InetAddress lServerAddr = InetAddress.getByName(SERVER\_IP);
```

Výpis 37: Formát InetAdress

Nyní se vytvoří a inicializuje objekt třídy *Socket*. Parametry jsou IP adresa zařízení, ke kterému se mobilní zařízení má připojit, a číslo portu.

```
Socket socket = new Socket(serverAddr, SERVER_PORT);
```

Výpis 38: Inicializace socketu

Poté je potřeba inicializovat již dříve vytvořené zásobníky *BufferOut* a *BufferIn* s využitím socketu. Pro správné formátování paketů je nutno definovat znakovou sadu ISO_8859_1.

```
mBufferOut = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.
    getOutputStream(), StandardCharsets.ISO\_8859\_1)), true);

mBufferIn = new BufferedReader(new InputStreamReader(socket.getInputStream(),
    StandardCharsets.ISO\_8859\_1));
```

Výpis 39: Inicializace výstupního a vstupního zásobníku

K zápisu zprávy se používá výstupní zásobník *BufferOut* a jeho metoda *write*. Zde je parametrem jeden prvek ze zasílané zprávy. V tomto případě se jedná o integer proměnnou. Tímto způsobem se naplní zásobník a metodou *flush* se naplněný zásobník pošle.

```
for (int i: aInput)
   mBufferOut.write(i);
```

```
mBufferOut.flush();
```

Výpis 40: Naplění a zaslání zásobníku

Pro čtení se použije vstupní zásobník *BufferIn* a jeho metoda *read*. Zde je vstupním parametrem ukazatel na pole znaků. Po úspěšném přečtení je pole naplněno zprávou a funkce vrací délku zprávy, tedy počet bytů.

```
int lBytes = mBufferIn.read(lBuffer);
```

Výpis 41: Čtení vstupního zásobníku

K ukončení spojení slouží metoda *close*.

```
socket.close();
```

Výpis 42: Ukončení TCP spojení

Tyto metody byly implementovány do třídy *TcpThread*, která dědí vlastnosti třídy Thread. Tím se objekt této třídy stává dalším vláknem. To se po vytvoření instance spouští metodou *start*.

```
tcpThread = new TcpThread();
tcpThread.start();
```

Výpis 43: Spuštení vlákna pro TCP komunikaci

6.3.3 Jednotná komunikační třída

Pro snadnější obsluhu komunikace je vhodné sloučit funkce obou komunikací do jedné třídy, kde není potřeba neustále zjišťovat, který typ komunikace je právě používán. K tomu slouží třída *Connection*.

Pro připojení jsou určeny metody *ConnectTcp* a *ConnectBluetooth*.

```
public boolean ConnectTcp();
public boolean ConnectBluetooth();
```

Výpis 44: Připojení

K zaslání zprávy slouží metoda *Write*, kde parametrem je pole typu integer. Pro čtení je určená funkce Read, která po přečtení zprávy vrací pole typu integer.

```
public void Write(int[] aInput);
public int[] Read();
```

Výpis 45: Zápis a čtení

6.4 Zpracování dat a vizualizace

Řešení režimu pro zpracování dat a vizualizaci je složeno z několik částí. Dohromady dávají dostačující možnosti uživateli pro zobrazení dat, jak textovou formou, tak grafickou formou.

6.4.1 Výpis dat

Nejprve je důležité zajistit vstupní parametry, které udávají, jaká data se budou číst. Prvním takovým parametrem je číslo uzlu. Z popisu ukládání dat do databáze v předchozí kapitole je známo, že pro každý existující uzel v IQMESH síti existuje jedna tabulka záznamů. To znamená, že tímto parametrem se určuje, ze které tabulky se bude číst.

Dalším parametrem je časové rozmezí. Každý záznam v tabulce jednotlivých uzlů obsahuje časové razítko. Rozmezí, které se tedy použije jako parametr, je porovnáváno právě s těmito časovými razítky.

Tlačítko *Číst* zpracuje všechny tyto parametry a využije příkaz *ReadLogsByTime*, který je součástí sady příkazů pro komunikační bránu. Po odeslání příkazu jsou jednotlivé záznamy ukládány. Po obdržení konečné potvrzovací zprávy se záznamy zapíší do textového pole podle definované tvaru. Tvar obsahuje číslo ID, měřenou hodnotu, jednotku (pokud je známá) a časové razítko.

6.4.2 Graf

Graf je dalším režimem celé aplikace. Využívá se zde prvek *GraphView*. Pro použití byla potřeba externí knihovna *GraphView*. Ta obsahuje několik podknihoven, které je potřeba importovat do programu aplikace.

```
import com.jjoe64.graphview.DefaultLabelFormatter;
import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.series.DataPoint;
import com.jjoe64.graphview.series.LineGraphSeries;
```

Výpis 46: Import pod knihoven pro práci s grafem

Pro fungování je zapotřebí nastavit několik parametrů pro správné zobrazení legend, popisů a samotných os.

Body pro zobrazení v grafu jsou definovány pomocí pole typu *LineGraphSeries<DataPoint>*. Takto vytvořené pole s body je následně přidáno do objektu grafu.

V této aplikaci je graf naplněn a zobrazen po úspěšném obdržení všech čtených dat z databáze. Spodní osa grafu znázorňuje časový průběh. Pro lepší přehlednost jsou popisy jednotlivých časů otočeny o 90 stupňů tak, aby se navzájem nepřekrývaly.

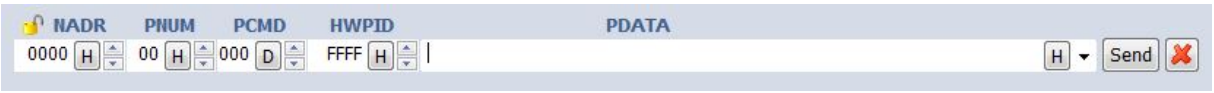
Vertikální osa zobrazuje měřenou hodnotu. Pokud je známá jednotka, je zobrazena spolu s typem dat. Celý grafický prvek *GraphView* je umístěn v horizontálně orientovaném prvku *ScrollView*, tak aby při větším množství dat bylo možno zobrazit všechna data.

6.4.3 Uložení dat

Nejprve je vytvořen soubor pomocí objektu třídy *File*. Při jeho inicializaci je definována cesta a název souboru. Složka, do které se záznamy ukládají, je *IQRF_zaznamy*. Umístěna je adresáři *ExternalStorage*. Název souboru je složený z čísla uzlu a aktuálního data. Datum je vytvořeno pomocí třídy *Date*. Při vytvoření objektu této třídy obsahuje objekt aktuální čas. Tento čas je poté zformátován objektem třídy *SimpleFormatter* do podoby *yyyy_MM_dd*. Pro uložení do souboru je zde použita třída *FileWriter*. Při vytváření objektu této třídy je v konstruktoru přidán objekt třídy *File*. Ten obsahuje cestu a název vytvořeného souboru. Pro přidání samotného textu jsou použity funkce *append* a *flush*. Soubor je poté zavřen funkcí *close*. Úspěšné zapsání je signalizováno vyskakovacím oknem.

6.5 DPA dotazování

Řešení DPA dotazování v aplikaci vychází z funkčního řešení v prostředí IQRF IDE. Základem je, aby mohl uživatel vložit do aplikace svůj DPA dotaz. Jak už bylo zmíněno v kapitole o IQRF, DPA dotaz se skládá z nejméně 7 bytů. Část PDATA nemusí dotaz vždy obsahovat, rovněž jeho délka může být různá. Z těchto kritérií se muselo vycházet při návrhu a následném řešení v aplikaci.



Obrázek 17: DPA dotazování v IQRF IDE

Jak je možno vidět v prostředí IQRF IDE, v části pro vkládání DPA dotazů se nachází několik vstupních kolonek rozdělených na NADR, PNUM, PCMD, HWPID a PDATA. Do buněk, kromě PDATA, je možno vpisovat jenom tak velká data, jaká může daná část dotazu obsahovat. U NADR to jsou 2 byty, u PNUM 1 byte, u PCMD 2 byty a u HWPID 2 byty. V části PDATA se musí jednotlivé byty oddělovat tečkou. Nastavování jednotlivých částí DPA dotazu probíhá v hexadecimálním tvaru. Prostředí nabízí přepnutí i do tvaru desítkového, avšak hexadecimální poskytuje lepší přehlednost.

IQRF IDE dále nabízí i možnost tzv. maker. Ta usnadňují vypisování DPA dotazu a základní dotazy se vyplní. Uživatel pak může měnit například jen adresu uzlu, tedy NADR. Prostředí nabízí i možnost nastavení vlastních maker.

Tyto hlavní funkce byly použity v aplikaci pro režim DPA dotazování. Jako textový vstup byl použit grafický prvek *EditText*. Jeho vstup lze nastavit pouze pro určitý formát, např. celá čísla, desetinná čísla nebo prostý text. V tomto případě je vstupem hexadecimální číslo, proto je vstup nastaven pro všechny znaky.

Část PDATA vychází rovněž z řešení v IQRF IDE, ale jednotlivé byty se od sebe oddělují dvojtečkou. Pro převod takto složeného textového řetězce do jednotlivých bytů bylo zapotřebí napsat funkci, která by řetězec rozdělila. Proto byla vytvořena funkce *PdataToBytes*.

```
public int[] PdataToBytes(String aInput);
```

Výpis 47: Převedení obsahu vstupu PDATA na byty

Na (Obr. **??**) lze vidět grafické uspořádání jednotlivých textových vstupů. Pro zpracování jednotlivých vstupů je zde tlačítko *Poslat DPA*.

Pro zobrazení odpovědí, které aplikace přijímá od komunikační brány, jsou vytvořeny dva textové výstupy prvkem *TextView*. Výstupy jsou rozmístěny jako dva sloupce horizontálně vedle sebe. První výstup slouží pro zobrazení typu zprávy. Druhý výstup je pro samotné DPA zprávy. Jelikož zprávy mohou být dlouhé až 64 bytů, bylo zapotřebí umístit druhý textový výstup do prvku *ScrollView* v horizontální orientaci. To umožní zobrazit celou zprávu. Oba textové výstupy byly umístěny do dalšího prvku *ScrollView* ve vertikální orientaci pro zobrazení většího počtu zpráv.

Jako předdefinovaná makra zde byla použita 3 tlačítka. Ta nastaví textové vstupy do daného tvaru a uživatel má možnost s nimi dále pracovat. Pro demonstrační účely byly použity dotazy pro získání teploty z integrovaného snímače na IQRF modulu a dva dotazy na rozsvícení a zhasnutí LED diody, umístěné taktéž na IQRF modulu.

Po stisknutí tlačítka *Poslat DPA* se vstupy zpracovávají a DPA dotazy jsou zasílány do komunikační brány pomocí metody *Write* z objektu třídy *Connection*. Dotazy jsou rovnou zapsány do textového pole a označeny jako Tx Request.

Jelikož se komunikace s komunikační bránou provádí v jiném vlákně, bylo zapotřebí použít tzv. *handler*. Ten umožnil zapisovat asychronně přijímané odpovědi do textových výstupů. Zde je provedeno i určení typu zprávy a poté je typ zapsán do textového pole.

6.6 Správa logování

V prvé řadě musí uživatel znát aktuální stav logování. Měl by vědět, zda je logování spuštěno nebo vypnuto a také, jaký je logovací interval. Pro zjištění stavu logování obsahuje tento režim tlačítko *Stav*. Pro zjištění stavu logování využívá program dotázání komunikační brány příkazem *GetLoggingState*.

Přijatá odpověď na tento dotaz obsahuje informaci o tom, zda je logování spuštěno, a podle toho vyplní textové pole textem "Spuštěno"nebo "Vypnuto".

Pro zjištění logovacího intervalu slouží tlačítko *Aktuální časový interval*. To využívá příkaz *GetLoggingInterval*. Jeho odpověď obsahuje časový interval v sekundách, který je zapsán do textového pole.

K zapnutí nebo vypnutí logování jsou určena tlačítka *Spustit* a *Zastavit*. Po stisknutí je využit příkaz *SetLoggingState* a jeho parametr *State* je při spuštění nastaven na hodnotu "1"a při vypnutí na hodnotu "0".

Pro nastavení logovacího intervalu slouží tlačítko *Nastavit*, které přečte textový vstup. Při zadání správné hodnoty je využit příkaz *SetLoggingInterval* a logovací interval v komunikační bráně je nastaven na hodnotu danou uživatelem.

6.7 Správa IQMESH sítě

Správa IQMESH sítě vychází z funkčního řešení z prostředí IQRF IDE, přesněji z nástroje IQMESH Network Manager. V aplikaci byly však použity jen základní, avšak dostačující, funkce.

Základní funkcí, která musí být uživateli přístupná pro správu IQMESH sítě, je funkce *Bond*. Parametrem pro tuto funkci je adresa, která bude přiřazena k nově připojenému uzlu. V aplikaci je tato funkce řešena tlačítkem *Bond* a vstupním textovým polem pro adresu uzlu. Funkce využívá zaslání DPA dotazu *Bond*. Po úspěšném provedení příkazu a obdržení odpovědi je prostřednictvím vyskakovacího okna zobrazen výsledek.

Další funkcí je funkce *Unbond*. Ta potřebuje pro vykonání parametr, který obsahuje číslo uzlu. Zde je řešení totožné s řešením funkce *Bond*. K provedení funkce využívá program DPA dotaz *Unbond*. Po úspěšném provedení příkazu a obdržení odpovědi je prostřednictvím vyskakovacího okna zobrazen výsledek.

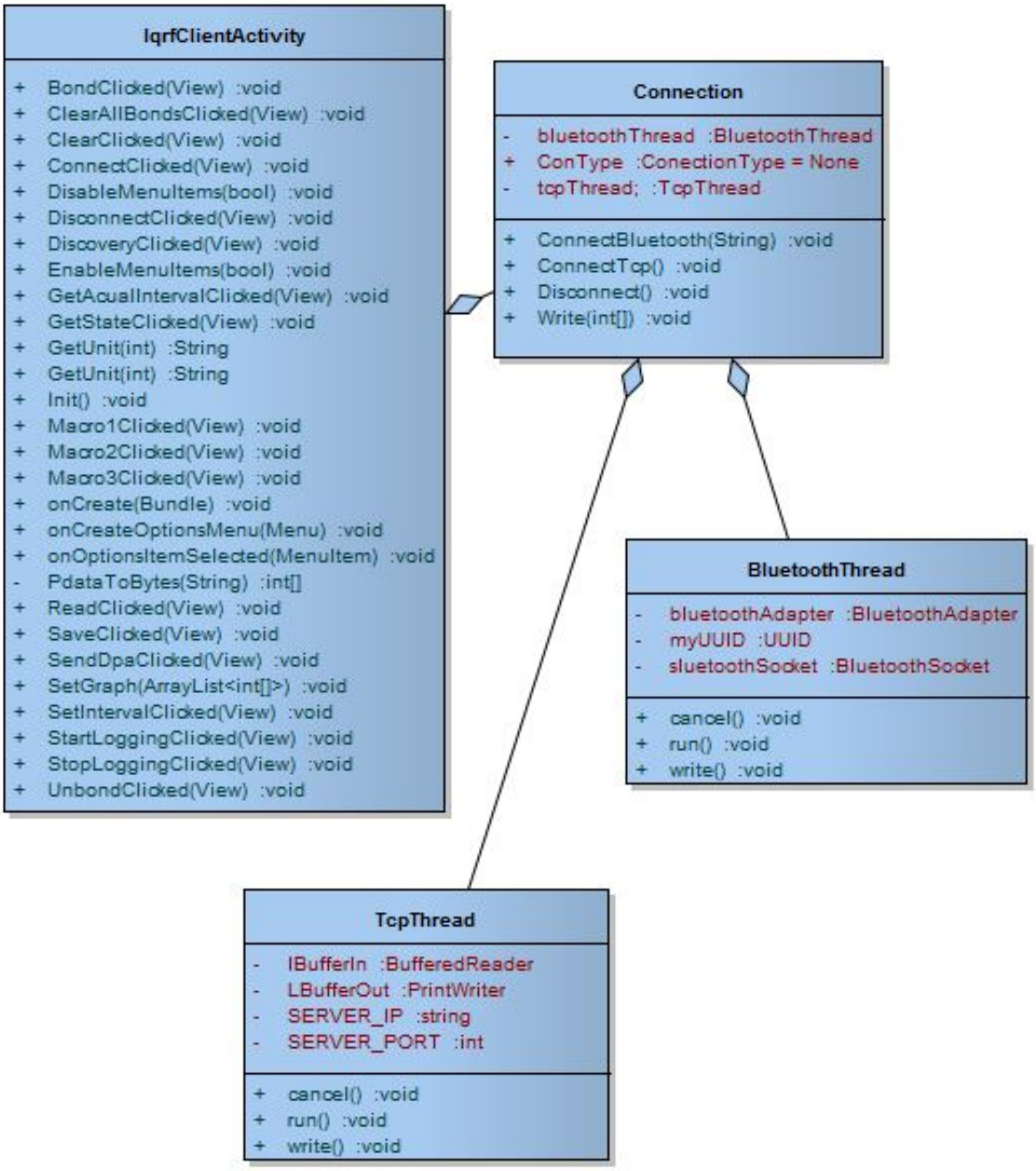
Pro vymazání všech spojení v IQMESH síti je zde funkce *Clear all bonds*. K nastavení této funkce slouží prvek *CheckBox*, jehož hodnota určuje, zda bude funkce *Clear all bonds* provedena pouze pro koordinátora sítě, nebo v celé IQMESH síti. Po úspěšném provedení příkazu a obdržení odpovědi je prostřednictvím vyskakovacího okna zobrazen výsledek.

Funkce *Discovery* využívá pro svou činnost 2 parametry. Prvním je vysílací výkon, neboli Tx výkon. Druhým parametrem je maximální adresa uzlu, do které bude funkce *Discovery* provedena. Po úspěšném provedení příkazu a obdržení odpovědi je prostřednictvím vyskakovacího okna zobrazen výsledek.

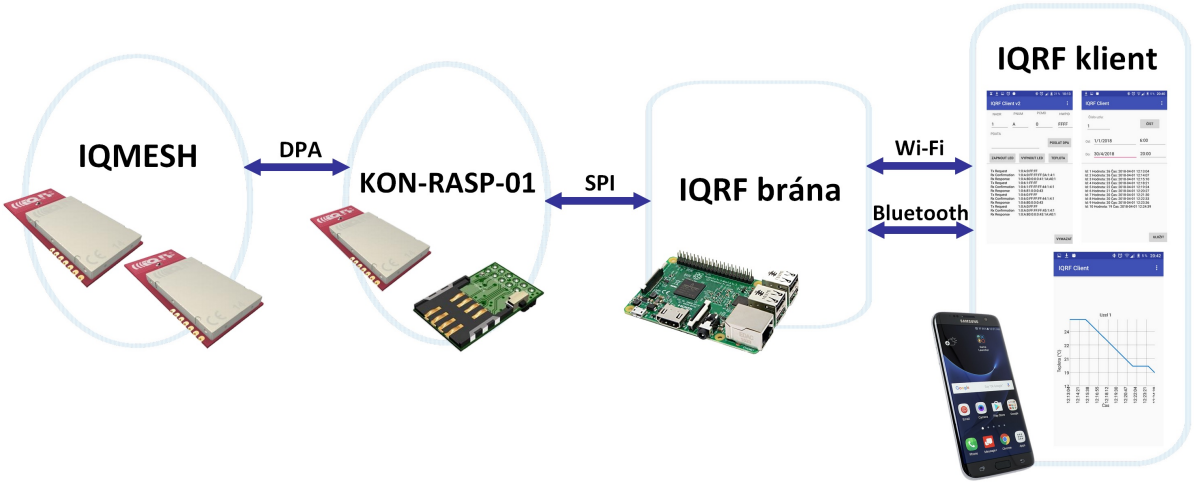
6.8 Výsledná struktura aplikace

Na Obr. 18 lze vidět diagram tříd výsledné mobilní aplikace.

Celkový systém je zobrazen v Obr. 19. Zahrnuje kompletní řešení od IQMESH sítě, fyzické připojení koordinátora k počítači Raspberry Pi 3 až po mobilní aplikaci.



Obrázek 18: Připojení

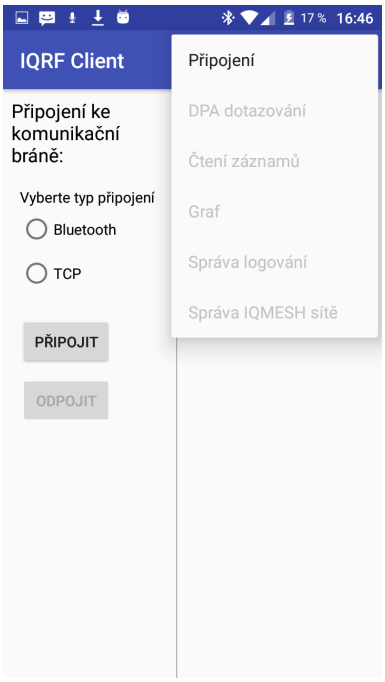


Obrázek 19: Struktura celkového systému

7 Zpracování a vizualizace měřených dat

7.1 Připojení mobilní aplikace ke komunikační bráně

Po spuštění aplikace se zobrazí režim pro připojení. V tuto chvíli není aplikace připojena ke komunikační bráně. Uživatel tedy nemá přístup k ostatním režimům, k nimž se lze dostat pomocí vysouvací nabídky (Obr. 20).

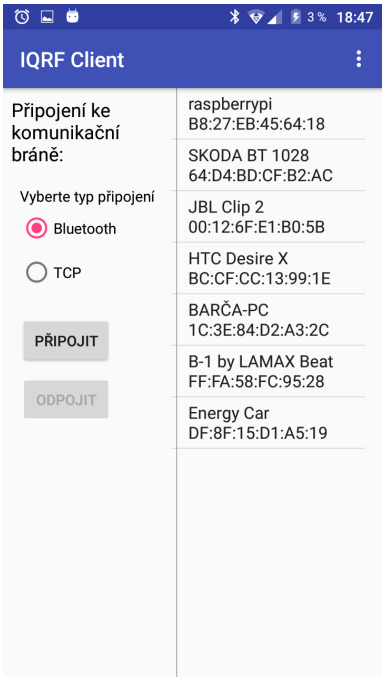


Obrázek 20: Nabídka není povolena

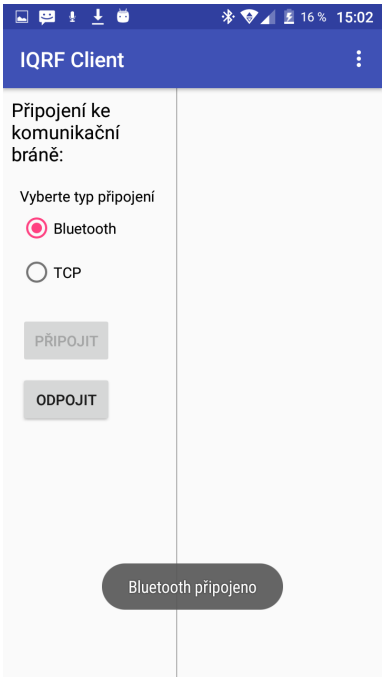
Prvním krokem při používání mobilní aplikace je připojení ke komunikační bráně. Jednou z možností je připojit se ke komunikační bráně pomocí technologie Bluetooth. Řešení vychází z předpokladu, že mobilní zařízení je s komunikační bránou spárováno, podle postupu zmíněného v kapitole popisující postup řešení programu komunikační brány. Pro připojení je nutné zvolit políčko *Bluetooth* v režimu pro připojení. Po kliknutí na tlačítko *Připojit* se v pravém sloupci objeví spárovaná zařízení (Obr. 21).

Pro připojení ke komunikační bráně je potřeba zvolit zařízení *raspberrypi*. Po úspěšném připojení se zobrazí vyskakovací okno, které signalizuje úspěšné připojení (Obr. 22).

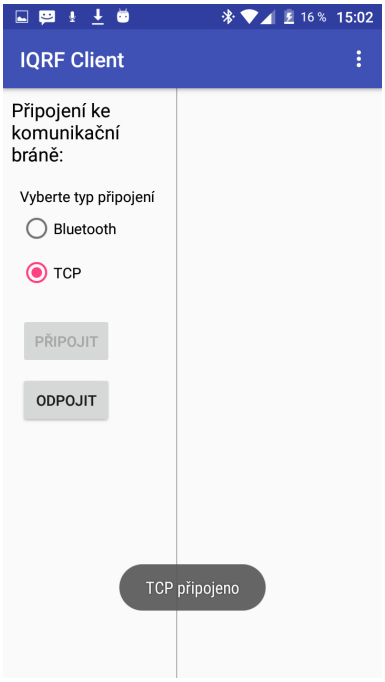
Pro připojení ke komunikační bráně pomocí TCP protokolu, který zastupuje technologii Wi-Fi, je zapotřebí zvolit políčko *TCP* a stisknout tlačítko *Připojit*. Zde se opět předpokládá, že komunikační brána, tedy Raspberry Pi 3, je nastavena jako Wi-Fi přístupový bod. Postup pro nastavení je sepsán v kapitole popisující postup řešení programu komunikační brány. Je také zapotřebí, aby mobilní zařízení bylo do této sítě připojeno. Po úspěšném připojení ke komunikační bráně se objeví vyskakovací okno, které signalizuje úspěšné připojení (Obr. 23).



Obrázek 21: Spárovaná zařízení

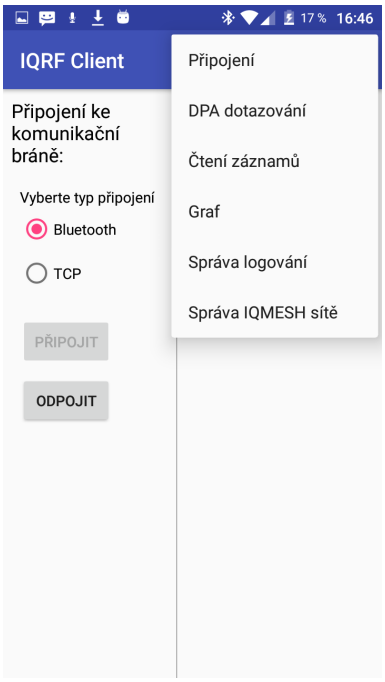


Obrázek 22: Připojení pomocí Bluetooth



Obrázek 23: Připojení pomocí TCP

V tomto bodě jsou uživateli zpřístupněny ostatní položky z nabídky a má tedy možnost přejít k hlavním funkcím aplikace. Zpřístupněnou nabídku lze vidět na Obr. 24.



Obrázek 24: Nabídka zpřístupněna

7.2 Přidání uzlů do IQMESH sítě

Pro přidání uzlů do IQMESH sítě slouží režim správy IQMESH. Zde má uživatel možnost přidávat a odebírat uzly z IQMESH sítě. V tomto případě se předpokládá, že do sítě není připojen žádný uzel.

Pro demonstrační účely budou připojeny dva IQRF moduly TR-72D. Ty jsou součástí vývojových kitů DK-EVAL-04a (Obr. 25). Ty disponují akumulátorem, dvěma tlačítky, signalizačními LED diodami a jednoduchým přístupem k periferiím modulu. Slouží k jednoduchým testovacím aplikacím.

Obrázek 25: DK-EVAL-04a

Pro přidání prvního uzlu je zapotřebí kliknout na tlačítko *Bond*. Po stiknutí koordinátor sítě čeká na potvrzení uzlu, který se chce do sítě připojit. Potvrzení se provede stisknutím tlačítka (blíže k USB konektoru) na vývojovém zařízení DK-EVAL-04a. Po stisknutí tlačítka se IQRF modul připojí a je mu přiřazena adresa. V tomto případě je adresa přidělována automaticky a modul má tedy adresu 1. Přiřazena adresa je zobrazena ve vyskakovacím okně (Obr. 26).

Stejný postup bude aplikován i na druhý modul. Zde bude přiřazena adresa 2 (Obr. 27).

Po připojení obou modulů je zapotřebí kliknout na tlačítko *Discovery*. Zde je možné nastavit parametry funkce, a to *Tx výkon* a maximální adresu, do které bude prováděna funkce *Discovery*. Po provedení funkce zjistí strukturu sítě. Ve vyskakovacím okně je zobrazen počet nalezených uzlů (Obr. 28).

7.3 Zasílání DPA dotazů a čtení odpovědí

Pro zaslání DPA dotazů do IQMESH sítě bude použit režim DPA dotazování. Zde je možnost sestavit vlastní DPA dotaz a pro rychlé otestování využít připravená makra. K testování DPA dotazování budou použita právě tato makra. Prvními dotazy budou tedy zapnutí a vypnutí červené LED diody, umístěné na modulu IQRF. Použité dotazy jsou zobrazeny v Tab. 25 a v Tab. 26.

Po odeslání dotazu se rozsvítí červená LED dioda na vývojovém zařízení DK-EVAL-04a, který je součástí uzlu 1. Na Obr. 29 lze vidět, že nejprve je přijata potvrzovací zpráva od

Obrázek 26: Přidání uzlu 1

Obrázek 27: Přidání uzlu 2

Obrázek 28: Zjištění struktury sítě

Tabulka 25: DPA dotaz pro zapnutí LEDR v uzlu 1

NADR	PNUM	PCMD	HWPID
1	6	1	FFFF

Tabulka 26: DPA dotaz pro vypnutí LEDR v uzlu 1

NADR	PNUM	PCMD	HWPID
1	6	0	FFFF

koordinátora sítě, tedy *Rx Confirmation*. Poté je přijata i odpověď od uzlu číslo 1, tedy *Rx Response*. Ta potvrzuje nastavení dané periférie.

Obrázek 29: Zapnutí a vypnutí LED diody na uzlu číslo 1

Stejný postup je aplikován i pro uzel číslo 2. Zde jsou použita opět makra pro zapnutí a vypnutí LED diody. Pro zaslání dotazu do uzlu 2 je zapotřebí změnit adresu, tedy HWPID, na hodnotu 2. Použité dotazy jsou zobrazeny v Tab. 27 a v Tab. 28.

Tabulka 27: DPA dotaz pro zapnutí LEDR v uzlu 2

NADR	PNUM	PCMD	HWPID
2	6	1	FFFF

Tabulka 28: DPA dotaz pro vypnutí LEDR v uzlu 2

NADR	PNUM	PCMD	HWPID
2	6	0	FFFF

Po odeslání dotazu se rozsvítí červená LED dioda na vývojovém zařízení DK-EVAL-04a, který je součástí uzlu 2. Průběh komunikace lze vidět na Obr. 30.

Jako další testovací dotaz je použit dotaz pro přečtení teploty. Použitý příkaz čte teplotu ze senzoru umístěného v modulu IQRF. Použitý dotaz je zobrazen v Tab. 29.

Obrázek 30: Zapnutí a vypnutí LED diody na uzlu číslo 2

Tabulka 29: DPA dotaz pro přečtení teploty z uzlu 1

NADR	PNUM	PCMD	HWPID
1	A	0	FFFF

Průběh komunikace je zobrazen na Obr. 31. Teplota je uložená ve 2 bytech v části *DPA value*.

Obrázek 31: Čtení teploty z uzlu číslo 1

7.4 Demonstrační měření a zobrazení naměřených dat

7.4.1 Popis měření

Tato podkapitola zahrnuje experimentální měření, které využívá funkce sběru dat, která je součástí komunikační brány. Pro toto měření jsou využity 2 IQRF moduly. Oba tyto moduly disponují vývojovým zařízením DK-EVAL-04a. Měření předpokládá, že oba moduly jsou jíž připojené do IQMESH sítě s komunikační bránou.

Měření má ověřit funkčnost sběru dat z IQMESH sítě s využitím rozpoznávání typů měřených dat podle HWPID. Tento princip je popsán v kapitole popisující řešení programu komunikační brány. Je tedy zapotřebí aby moduly obsahovaly Custom DPA Handler, ve kterém je definován HWPID, který určuje typ měřených dat.

Pro toto měření jsou vybrány dva typy měřených veličin, a to teplota a intenzita světla. Oba tyto typy jsou definovány v tabulce *HardwareProfilesList*. Teplota je tomto případě získávána z teplotního senzoru umístěného na IQRF modulu. Pro měření intenzity světla je využit fotorezistor. Ten je součástí senzorového zařízení DDC-SE-01 (Obr. 32, které je rozšířením DK-EVAL-04a.

Obrázek 32: DDC-SE-01

7.4.2 Nastavení logování

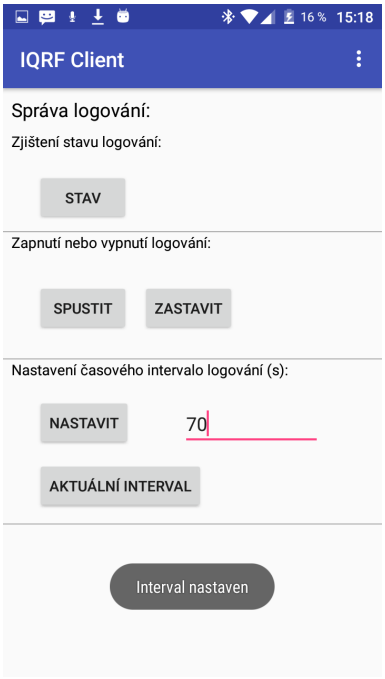
Nejdříve je potřeba nastavit interval logování. Ten se nastaví v režimu správy logování v mobilní aplikaci. Kliknutím na tlačítko Stav je zobrazen aktuální stav logování, tzn. zapnuté nebo vypnuté. Z Obr. 33 lze vidět, že logování je vypnuté.

Obrázek 33: Logování je vypnuto

Kliknutím na tlačítko *Aktuální interval* je zjištěn aktuálně nastavený logovací interval v komunikační bráně. Interval se zobrazí ve vyskakovacím okně Obr. 34

Pro nastavení jiného intervalu je zapotřebí interval zadat do textového pole. Interval je zadáván v sekundách. Poté je zapotřebí kliknout na tlačítko *Nastavit*. Nastavení je potvrzeno vyskakovacím oknem (Obr. 35).

Obrázek 34: Aktuální logovací interval



Obrázek 35: Změna intervalu

Spuštění logování se provede kliknutím na tlačítko *Spustit*. Po stisknutí je zobrazeno potvrzení změny a aktuální stav lze ověřit opětovným kliknutím na tlačítko *Stav* (Obr. 36).

Obrázek 36: Logování zapnuto

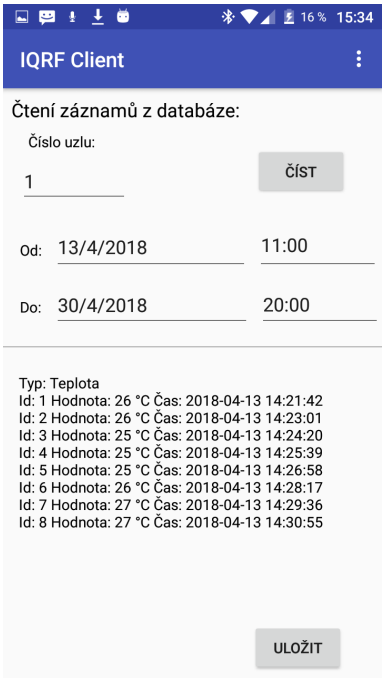
7.4.3 Zpracování a zobrazení naměřených dat

Měření probíhalo přibližně 10 minut. Nejprve byly moduly umístěny v místnosti s osvětlením. Poté byly přemístěny do venkovních podmínek. Pro zobrazení naměřených výsledků je zapotřebí přejít do režimu čtení záznamů. Zde je nutno vyplnit časové rozmezí, ve kterém měření probíhalo. Po kliknutí na tlačítko *Číst* se načtou naměřená data. V tomto případě se načtou záznamy pro uzel číslo 1 (Obr. 37). Lze vidět, že je načteno 8 záznamů. Dále je zobrazen typ měřených dat, ID, samotná hodnota, jednotka a čas záznamu.

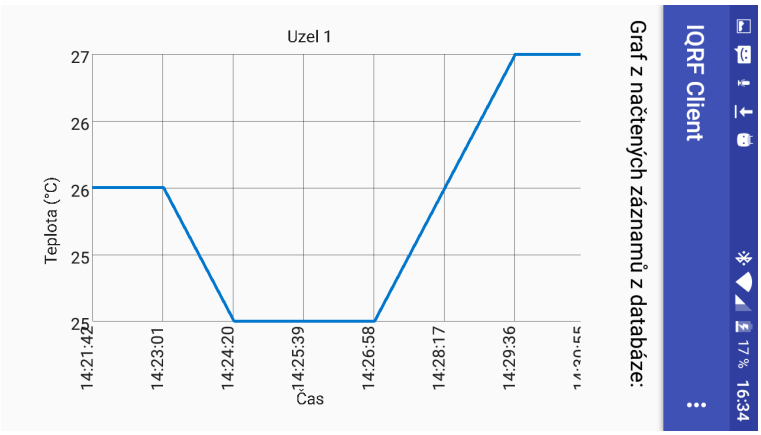
Po načtení jsou záznamy automaticky převedeny i do grafu. Ten se nachází pod položkou *Graf* ve vysouvací nabídce. Na grafu lze vidět horizontální osu, která zobrazuje čas a vertikální osu, zobrazující naměřené hodnoty (Obr. 38).

Stejný postup je aplikován i na druhý uzel. Je tedy nutné změnit číslo uzlu v režimu čtení záznamů. Po kliknutí na tlačítko *Číst* se zobrazí záznamy. Lze vidět, že se jedná o typ *Poměrná světelná intenzita* (Obr. 39).

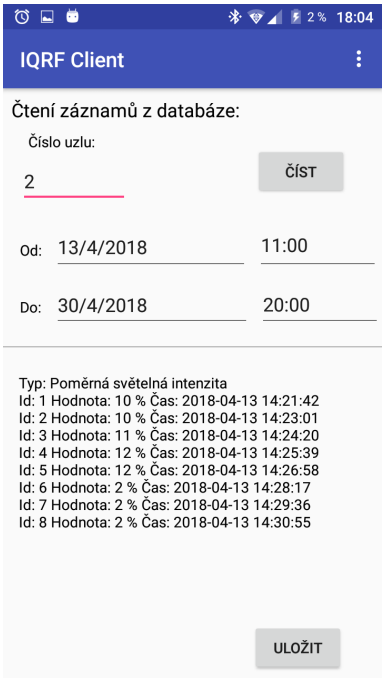
Záznamy jsou opět zobrazeny i v grafu (Obr. 40).



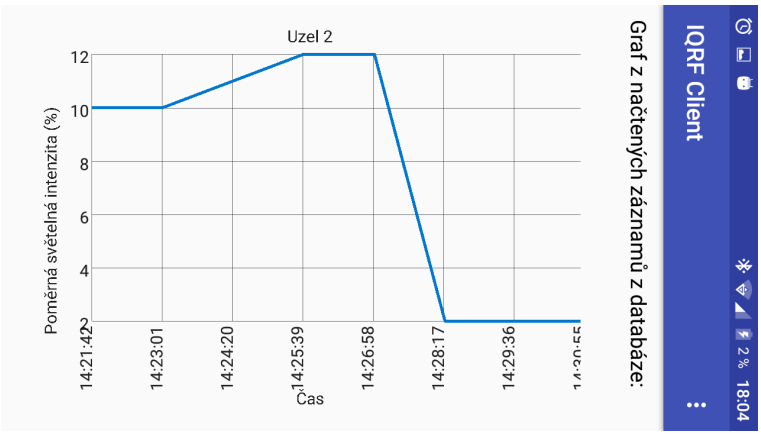
Obrázek 37: Načtené záznamy z databáze pro uzel číslo 1



Obrázek 38: Graf měření teploty



Obrázek 39: Načtené záznamy z databáze pro uzel číslo 2



Obrázek 40: Graf měření poměrné světelné intenzity

8 Závěr

Výsledkem této práce je systém skládající se ze dvou aplikací. Jednou z nich je komunikační brána, fungující na počítači Raspberry Pi 3. Aplikace zajišťuje několik funkcí. První funkcí je samotná brána, kdy aplikace přeposílá DPA pakety mezi IQMESH sítí a mobilní aplikací.

Další funkcí je sběr dat z IQMESH sítě a ukládání do databáze. Funkce umožňuje logování měřených dat z IQMESH sítě v nastavitelném intervalu. Jednotlivé uzly jsou identifikovány podle HWPID, určujícího typ měřených dat. Měřená data jsou pak ukládána do databáze ve správném formátu. Další funkcí je zasílání naměřených dat do mobilní aplikace, odkud je možné i samotné logování spravovat. Ke komunikační bráně se lze připojit pomocí technologií Bluetooth a Wi-Fi.

Druhou částí řešení je mobilní aplikace určená pro operační systém Android. Ta umožňuje uživateli připojit se ke komunikační bráně pomocí vybrané komunikační technologie. Poté může uživatel zasílat a číst příchozí DPA pakety prostřednictvím komunikační brány.

Dále je umožněno načtení záznamů získaných funkcí sběru dat a uložených v databázi v komunikační bráně. Tyto záznamy lze zobrazit ve formě výpisu a ve formě grafu. Pro další zpracování je možnost data uložit do textového souboru. V aplikaci lze nastavit logovací interval a logování spustit nebo zastavit. V aplikace uživatel dále může spravovat IQMESH sítě. Zde je umožněno připojit nové uzly a dále, pomocí více funkcí, uzly odpojovat.

Celkový systém má praktické využití v případech, kde je potřeba lokální přístup do IQMESH sítě. Pomocí režimu logování pak lze v této síti provádět dlouhodobá experimentální měření s možností dalšího zpracování naměřených dat. Systém tak má využití jak v průmyslových aplikacích, tak i při statistických měřeních nebo v aplikacích typu chytrých domů.

Z celkového systému lze vyjít i při dalším vývoji, a to zejména v oblasti mobilní aplikace, kde je možné navrhnout jednoúčelovou aplikaci, která vychází z tohoto funkčního řešení. To znamená navrhnout takovou aplikaci, která je svými vizualizačním a ovládacím zpracováním přizpůsobena danému použití, např. aktuální zobrazování měřených hodnot nebo ovládání chodu zařízení.

Literatura

[1] VÁŇA, Martin. Návrh a ověření rádiové sítě typu MESH pro bez licenční pásmo. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně.

[2] Bezdrátová inovace pro malá data jménem IQRF. Root.cz [online]. 2016, 11. 5. 2016 [cit. 2018-01-24]. Dostupné z: https://www.root.cz/clanky/bezdratova-inovace-pro-mala-data-jmenem-iqrf/

[3] TR-72D series. IQRF Technology [online]. [cit. 2018-02-4]. Dostupné z: https://www.iqrf.org/products/transceivers/tr-72d

[4] IQRF DPA Framework - Technical Guide. IQRF Technology [online]. [cit. 2018-04-28]. Dostupné z: https://www.iqrf.org/DpaTechGuide/

[5] STRAKA, Jaroslav. IoT a IBM Bluemix. ITBiz [online]. 2017, 19. leden 2017 [cit. 2018-03-28]. Dostupné z: http://www.itbiz.cz/clanky/iot-a-ibm-bluemix

[6] Accessing The Database. Raspberry Pi Resources [online]. [cit. 2017-12-5]. Dostupné z: http://www.raspberry-projects.com/pi/programming-in-c/databases-programming-in-c/mysql/accessing-the-database

[7] RFCOMM sockets. An Introduction to Bluetooth Programming [online]. [cit. 2017-11-2]. Dostupné z: http://people.csail.mit.edu/albert/bluez-intro/x502.html

[8] Sockets Tutorial. Www.LinuxHowtos.org [online]. [cit. 2018-12-2]. Dostupné z: http://www.linuxhowtos.org/C_C++/socket.htm

[9] Protokol TCP - I. EArchiv.cz [online]. [cit. 2018-04-28]. Dostupné z: http://www.earchiv.cz/a93/a305c110.php3

[10] Raspberry Pi - Hotspot/Access Point dhcpcd method. Raspberry Connect [online]. 2017, 19. 11. 2016 [cit. 2018-04-01]. Dostupné z: http://www.raspberryconnect.com/network/item/333-raspberry-pi-hotspot-access-point-dhcpcd-method

[11] HÁK, Jan. Vozítko založené na Raspberry Pi. Plzeň, 2016. Bakalářská práce. Západočeská univerzita v Plzni.

[12] HÜBNER, Pavel. Systém sběru dat s Raspberry Pi pro domovní automatizaci. Praha, 2015. Bakalářská práce. České vysoké učení technické v Praze. Vedoucí práce Jan Fischer.

[13] Raspberry Pi 3 Model B. Raspberry Pi [online]. [cit. 2017-11-6]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[14] ČÁNSKÝ, Jiří. Bluetooth. Praha, 2006. Semestrální práce. České vysoké učení technické v Praze.

[15] Základy technologie Bluetooth: původ a rozsah funkcí. PCWorld [online]. 2009, 10.2.2009 [cit. 2018-03-23]. Dostupné z: https://pcworld.cz/hardware/Zaklady-technologie-Bluetooth-puvod-a-rozsah-funkci-6635

[16] TRČÁLEK, Antonín. Všechno, co byste měli vědět o Wi-Fi. Živě [online]. 2012, 14.3.2012 [cit. 2018-03-02]. Dostupné z: https://www.zive.cz/clanky/vsechno-co-byste-meli-vedet-o-wi-fi/sc-3-a-162796/default.aspx

[17] Bezdrátové sítě WiFi - 2. Díl Principy a pravidla Wireless LAN. PCWolrd [online]. 2003, 1.1.2003 [cit. 2017-11-4]. Dostupné z: https://pcworld.cz/internet/bezdratove-site-wifi-2-dil-principy-a-pravidla-wireless-lan-13405

[18] Azure IoT Suite. Microsoft [online]. [cit. 2018-1-21]. Dostupné z: https://www.microsoft.com/cs-cz/internet-of-things/azure-iot-suite

A Příloha

Přiložené CD obsahuje elektronickou verzi bakalářské práce a zdrojové kódy.